40347

National Library of Canada

Bibliothèque nationale du Canada

CANADIAN THESES ON MICROFICHE

THÈSES CANADIENNES SUR MICROFICHE

NAME OF AUTHOR/*NOM DE L'AUTEUR* K. MICHAEL WEISS

TITLE OF THESIS/*TITRE DE LA THÈSE* A MICROCOMPUTER DEVELOPMENT SYSTEM FOR MACHINE CONTROL APPLICATIONS

UNIVERSITY/*UNIVERSITÉ* UNIVERSITY OF ALBERTA

DEGREE FOR WHICH THESIS WAS PRESENTED/ *GRADE POUR LEQUEL CETTE THÈSE FUT PRÉSENTÉE* M. Sc.

YEAR THIS DEGREE CONFERRED/*ANNÉE D'OBTENTION DE CE GRADE* 1978

NAME OF SUPERVISOR/*NOM DU DIRECTEUR DE THÈSE* DR. D. H. KELLY, PROF. P. J. HARDING

Permission is hereby granted to the NATIONAL LIBRARY OF CANADA to microfilm this thesis and to lend or sell copies of the film.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

L'autorisation est, par la présente, accordée à la BIBLIOTHÈQUE NATIONALE DU CANADA de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans l'autorisation écrite de l'auteur.

DATED/*DATÉ* Oct 20/78 SIGNED/*SIGNÉ* Michael Weiss

PERMANENT ADDRESS/*RÉSIDENCE FIXE*
BOX 319 - SUB 11
UNIVERSITY OF ALBERTA
EDMONTON ALBERTA, T6G 2E0

NL-91 (3-74)

National Library of Canada

Cataloguing Branch
Canadian Theses Division

Ottawa, Canada
K1A 0N4

Bibliothèque nationale du Canada

Direction du catalogage
Division des thèses canadiennes

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us a poor photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de mauvaise qualité.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE

NL-339 (3/77)

THE UNIVERSITY OF ALBERTA

A Microcomputer Development System for Machine Control Applications

by

© K. Michael Weiss

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE
OF Master of Science
IN
Electrical Engineering

Department of Electrical Engineering

EDMONTON, ALBERTA
Fall 1978

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled A Microcomputer Development System for Machine Control Applications submitted by K. Michael Weiss in partial fulfilment of the requirements for the degree of Master of Science in Electrical Engineering.

D H Kelly

Supervisor

Pat Harding

Supervisor

Keith A. Stromsmoe

A W Peterson

Date Oct 5, 1978

to my family

Abstract

A microprocessor based DC motor speed control system is presented, incorporating digital proportional and integral control schemes in the system software. A variable duty cycle SCR/diode bridge is used to provide the DC controlling voltage for the DC motor aramture. RPM feedback is provided by a shaft mounted slotted disc and an optical interrupter circuit. A current to voltage transducer and an A/D converter circuit allow software monitoring of the armature current; providing instantaneous and I^2t current limiting.

A keyboard monitor system permits the modification of motor control constants and set points; as well as allowing the user to enter and execute assembly language programs.

Binary floating point addition, subtraction, multiplication, and division routines are included in the system software package. These routines are used in the implementation of the motor control software, as well as allowing the microprocessor system to be used as a four function calculator. Floating point decimal to binary and binary to decimal conversion routines are included to permit interfacing between the system and its human operator.

This work discusses the digital control algorithms used, and illustrates the motor's response to step load changes and RPM setpoint changes with various combinations of control parameters. The hardware and software of which the system is composed is also discussed.

Acknowledgements

The author greatfully acknowledges the encouragement and assistance of Professors D. H. Kelly and P. F. Harding in the preparation of this thesis, and in the related work. Thanks are also due to fellow graduate students Laurie Davis and Martin Nohr for their assistance, along with many of the members of the Department of Electrical Engineering.

The author greatfully acknowledges the financial support rendered by the National Research Council of Canada and the Department of Electrical Engineering of the University of Alberta.

Table of Contents

Introduction 1

  The Scope of this Thesis 2

Digital DC Motor Control 5

  The Motor Control Algorithm 6

    Digital Proportional Control 7

    Digital Integral Control 7

    The Control Output 8

  Linearization of the Output Voltage 9

  Calculation of System RPM 11

  Over Current Protection 12

System Test Results 15

  DC Motor with No Control 16

  DC Motor with Proportional Control 16

    Speed Load Characteristics at 1000 RPM 22

  Proportional Plus Integral Control 25

  The Application of a Step Load 30

  Changes in the RPM Setpoint 31

  The Effects of Changing the Sample Interval 31

  Saturating and Non-Saturating Arithmetic 36

  Operation at 1800 RPM 39

    Speed Load Characteristic at 1800 RPM 41

  Steady State Error with PI Control 41

Keyboard Operation of the Microcomputer System 43

  The Use of the Monitor System 44

    System Reset 44

        Data Entry Keys 45
        Load Memory Address 46
        Memory Read 46
        Memory Write 47
        The Run Function 48
        I/O Port Read 49
        I/O Port Write 49
        Auxiliary Functions 50
    Data Manipulation Routines 51
        Data Entry 51
        Arithmetic Operations 53
        Decimal to Binary Conversion 54
        Exchange X and Y Registers 54
        Storing and Recalling Data 55
    Operation of the DC Motor Control System 56
        Motor Control Routine Initialization 57
        Starting the Motor 58
        Stopping the Motor 59
        Altering and Reading Control Variables and Set Points 59
        Continuous Monitoring of Variables 60
        Monitoring the Power Mains Frequency 62
    System Error Messages 62
System Hardware 65
    Bus Structure 66
    AC Wiring 69
    System Power Supplies 70
        System Cooling 70

Microprocessor Board 72
    CPU Control Signals 72
    Address and Data Lines 77
Keyboard and Display Interface 77
Read Only Memory 84
Random Access Memory 85
Motor Interface Module 86
    The Front Panel LED Array 88
The Programmable SCR Circuit 89
    The Motor Field Supply 94
    Application Module Power Supplies 94
Armature Current Sensor 95
    The Current Sense Amplifier and Filter 96
    The Sample and Hold Circuit 98
    Analog to Digital Converter Circuitry 99
Optical Tachometer 100
Software Documentation 101
    Programming Conventions 101
The Keyboard Monitor 103
    Interrupt Vectors 104
    Error Exits 105
    System Reset 105
    The Keyboard Monitor Routine 106
    The Display Clear Routine 109
    Data Entry Routines 109
    Load Address Routine 110
    Reading and Displaying a Memory Location 110

- The Memory Write Function 111
- The Memory Read Function 111
- The Run Function 112
- Setting the Function Flag 112
- The I/O Port Read Function 112
- Port Data Display 113
- The I/O Port Write Function 113
- Hexadecimal to Seven Segment Code Conversion . . 114
- The Idle Loop 114
- The Data Block Move Routine 114

The Floating Point Arithmetic Package 115

- Binary Floating Point Number Format 115
- Floating Point Multiplication 115
- Sixteen Bit Binary Multiplication 119
- The Normalization Routine 120
- Floating Point Division 121
- Floating Point Addition 122
- Floating Point Subtraction 123
- The Four Byte Shift Routines 123
- Floating Point Decimal to Binary Conversion . . 123
- Floating Point Binary to Decimal Conversion . . 125

Calculator Routines 127

- The Y Register Load Function 127
- The Decimal Data Entry Routine 128
- The Decimal Data Display Routine 129
- The Change Sign Function 130
- Decimal to Hexadecimal Conversion 130

X and Y Register Exchange 130

Calculator Multiplication, Division, Addition and Subtraction Functions 131

The Calculator Function Start Routine 131

The Calculator Function End Routine 132

The Floating Point Workspace Save Function . . . 132

The Floating Point Workspace Restore Routine . . 132

The Data Store Function 133

The Data Recall Function 133

Motor Control Software 134

The Application Module Initialization Routine . 134

The Motor Start Function 135

Motor Control Update 135

The Current Test Routine 136

Calculation of the Control Output 136

The Control Voltage Output Routine 137

The Motor Stop Routine 138

The RPM Conversion Routine 138

The SCR Current Conversion Routine 139

Optical Tachometer Interrupt Service Routine . . 140

The Line Frequency Monitor Routine 140

Summary and Conclusion 143

Bibliography 145

Appendix A System Software Listing 149

Address and Label Definitions 149

Monitor Data File 149

Application Program Data File 151

I/O Port Addresses 151

The Monitor System 152

Interrupt Vectors 152

Forwarding Addresses 153

Interrupt Error Routine 153

Error Exit Routine 153

System Reset Routine 154

The Keyboard Interrupt Service Routine 155

The Display Clear Routine 159

Data Entry Routines 159

The Load Address Routine 161

The Memory Write Function 161

Reading and Displaying a Memory Location 162

The Memory Read Function 163

The Run Function 163

Setting the Function Flag 163

The I/O Port Read Function 164

I/O Port Data Display Routine 164

The I/O Port Write Function 165

Conversion from Hexadecimal to Seven Segment Code 166

The Idle Loop 166

The Data Block Move Routine 167

The Floating Point Arithmetic Package 168

Forwarding Addresses 168

Floating Point Binary Multiplication 168

Sixteen Bit Binary Integer Multiplication . . . 171

The Floating Point Normalization Routine 172

Floating Point Division 174

Floating Point Addition 178

Floating Point Subtraction 182

Four Byte Shift Left 182

Four Byte Shift Right 183

Forwarding Addresses 184

Floating Point Decimal to Binary Conversion . . 184

Floating Point Binary to Decimal Conversion . . 187

Calculator Function Routines 194

Forwarding Addresses 194

The Y Register Load Function 194

The Decimal Data Entry Routine 195

The Decimal Data Display Routine 196

The Sign Change Function 198

The Decimal to Hexadecimal Conversion Function . 198

The X and Y Register Exchange Function 199

The Multiplication Function 199

The Division Function 200

The Addition Function 200

The Subtraction Function 200

The Calculator Function Start Routine 201

The Calculator Function End Routine 201

The Data Store Function 201

The Data Recall Function 202

The Floating Point Workspace Save Routine . . . 203

The Floating Point Workspace Restore Routine . . 204

Application Module Software 205

Forwarding Addresses 205

The Application Module Initialization Routine . 205

The Motor Start Function 207

The Motor Control Update Routine 208

The Current Test Routine 209

Calculation of Control Output 211

Output of Control Voltage 213

The Motor Stop Routine 216

Forwarding Addresses 216

Line Frequency Monitoring Routine 216

Optical Tachometer Interrupt Service Routine . . 218

RPM Reading and Conversion 219

Reading and Conversion of SCR Current 220

List of Figures

Figure Page

1. Block Diagram of the DC Motor Speed Controller. . . 3
2. DC Motor Control Loop 6
3. Variable Duty Cycle, Full Wave Rectified Sine Wave. 10
4. Continuous Current vs. Allowed Time. 13
5. Speed Load Characteristic with No Control 17
6. No Load Start; No Control 17
7. No Load Start; Kp = 0.005 18
8. No Load Start; Kp = 0.02 18
9. No Load Start; Kp = 0.05 19
10. No Load Start; Kp = 0.1 19
11. No Load Start; Kp = 0.5 20
12. Speed Load Characteristic at 1000 RPM 21
13. No Load Start; Kp = 0.002, Ki = 0.01 23
14. No Load Start; Kp = 0.01, Ki = 0.01 23
15. No Load Start; Kp = 0.05, Ki = 0.01 24
16. No Load Start; Kp = 0.01, Ki = 0.001 24
17. No Load Start; Kp = 0.01, Ki = 0.04 25
18. 40 Watt Load Step; Kp = 0.005 27
19. 40 Watt Load Step; Kp = 0.02 27
20. 40 Watt Load Step; Kp = 0.05 27
21. 40 Watt Load Step; Kp = 0.1 28
22. 40 Watt Load Step; Kp = 0.002, Ki = 0.01 28
23. 40 Watt Load Step; Kp = 0.01, Ki = 0.01 28
24. 40 Watt Load Step; Kp = 0.05, Ki = 0.01 29
25. 40 Watt Load Step; Kp = 0.01, Ki = 0.001 29

26. 40 Watt Load Step; Kp = 0.01, Ki = 0.04 29

27. RPM Setpoint Change; Kp = 0.005 32

28. RPM Setpoint Change; Kp = 0.02 32

29. RPM Setpoint Change; Kp = 0.05 32

30. RPM Setpoint Change; Kp = 0.1 33

31. RPM Setpoint Change; Kp = 0.002, Ki = 0.01 33

32. RPM Setpoint Change; Kp = 0.01, Ki = 0.01 33

33. RPM Setpoint Change; Kp = 0.05, Ki = 0.01 34

34. RPM Setpoint Change; Kp = 0.01, Ki = 0.001 34

35. RPM Setpoint Change; Kp = 0.01, Ki = 0.04 34

36. 200 msec sample interval, Kp = 0.01, Ki = 0.02 . . 35

37. 250 msec sample interval, Kp = 0.02, Ki = 0.025 . . 36

38. Nonsaturating Arithmetic; Kp = 0.05, Ki = 0.01 . . 37

39. Saturating Arithmetic; Kp = 0.05, Ki = 0.01 37

40. No Load Start at 1800 RPM; Kp = 0.05, Ki = 0.02 . . 40

41. Speed Load Characteristic at 1800 RPM 41

42. RMS Velocity Error Under Typical Operating Conditions 42

43. System Keyboard Layout. 45

44. Motor Control Variables and Set Points. 57

45. Routine for Continuous Memory Register Display. . . 61

46. AC Wiring Diagram 69

47. Positive 12 Volt Regulator Circuit. 71

48. Negative 5 Volt Regulator Circuit 71

49. Positive 5 Volt Regulator Circuit 71

50. Main CPU Board - Control Signals. 73

51. Main CPU Board - Address and Data Lines 74

52. CPU Board IC's. 75

53. Peripheral IC Interface to System Bus 78

54. System Display and Keypad 79

55. Display Board IC's. 80

56. Read Only Memory Board. 81

57. ROM Board ICs 82

58. Random Access Memory Board. 83

59. RAM Board ICs 84

60. Application Module Interface Board - Address and Data Buffers 86

61. Application Module Interface ICs. 87

62. Front Panel LED Array 89

63. Motor Control Circuitry 90

64. Armature Voltage and Current - 50% Duty Cycle . . . 91

65. Armature Voltage and Current - 25% Duty Cycle . . . 91

66. Application Module Power Supply 93

67. Armature Current Sense Amplifier. 95

68. Analog to Digital Converter Interface 97

69. Optical Tachometer System 100

70. Memory Map - Read Only Memory 102

71. Memory Map - Random Access Memory 102

72. Memory Map - I/O Ports. 104

73. Keyboard Service Routine Flowchart. 108

74. The Floating Point Multiplication Routine 118

75. Optical Tachometer Interrupt Service Routine. . . 141

I. Introduction

The advent of microprocessor technology at a reasonable cost has opened the door to many dedicated computer control applications. One of these is the digital control of a synchronous generator and its prime mover. Direct digital control of a static excitation system allows measures to be taken to counteract system instabilities in a time frame significantly shorter than the time constant of the prime mover. The ability to make rapid adjustments of the generator output allows the extension of the stable operating region of long transmission lines, as well as permitting the damping of oscillations on intersystem ties [1-8]. Many methods of direct excitation control have been used, with varying degrees of success [9-24]. Today's availability of microprocessor technology has added another means for high speed excitation control.

The general optimal solution to the exciter control problem [25-28] would require too much calculation time in a normal MOS structure microprocessor configuration. However, representing the generator by a slightly simplified mathematical model, results in a set of equations that is close enough to the optimal solution for all practical purposes, yet is simple enough for real time calculation. The functionality of this model has been verified using computer models to compare the simplified model to a more detailed version [29].

A. The Scope of this Thesis

This work covers the design, construction, programming and testing of a digital machine control development system. The system is configured as a DC motor speed controller, but is designed with the addition of excitation control in mind. It is on its own a valuable tool for gaining an understanding of digital motor control and some of the associated considerations. The system consists of the following modules:

1. An expandable microcomputer system including read only memory and random access memory.
2. A keyboard and display to allow interaction between the operator and the microcomputer system.
3. An interface to link the microcomputer system to the application module, which includes:
    - a. An SCR based DC voltage source suitable for the excitation of a motor armature or a generator field. The duty cycle of the SCR circuit is program controllable.
    - b. A current to voltage transducer and A/D converter circuit to allow the microcomputer to read the motor armature or generator field current produced by the SCR circuit.
    - c. An optical tachometer; consisting of a shaft mounted slotted disc, along with the associated electronics required to produce latched interrupts as the disc

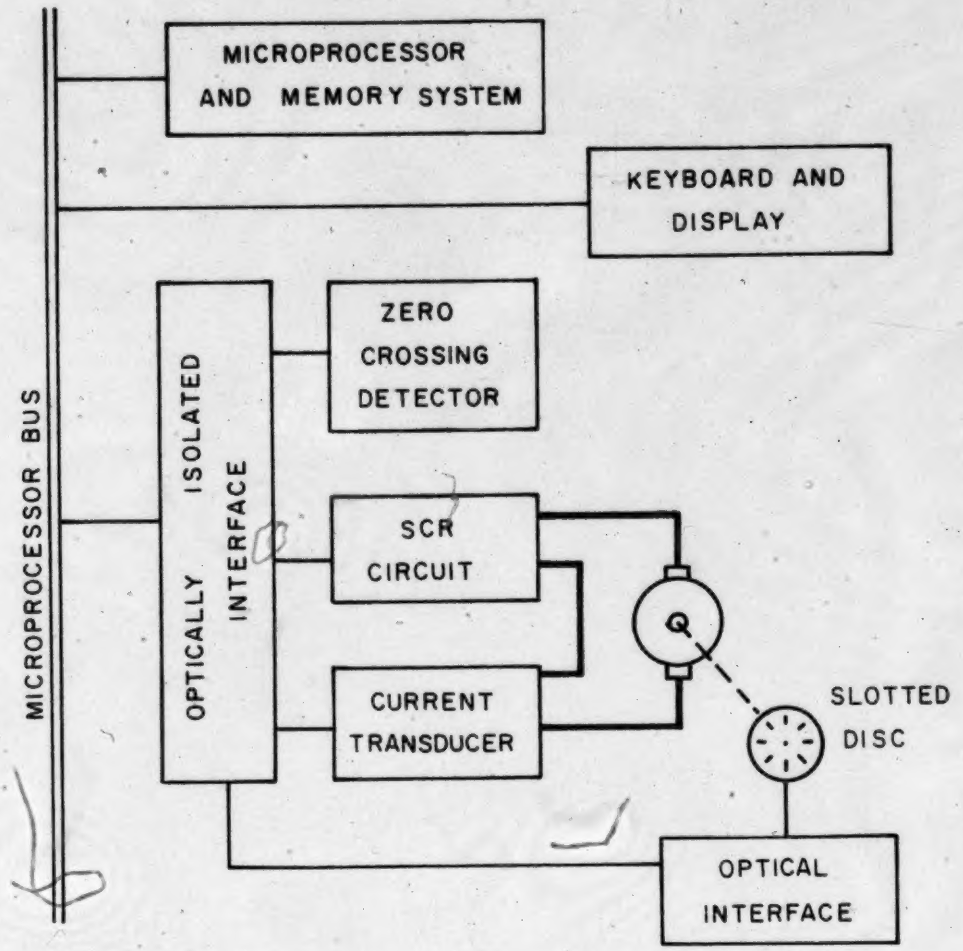

MICROPROCESSOR BUS
MICROPROCESSOR AND MEMORY SYSTEM
KEYBOARD AND DISPLAY
OPTICALLY ISOLATED INTERFACE
ZERO CROSSING DETECTOR
SCR CIRCUIT
CURRENT TRANSDUCER
SLOTTED DISC
OPTICAL INTERFACE

Figure 1 Block Diagram of the DC Motor Speed Controller

rotates.

d. A zero crossing detector circuit which produces a synchronizing pulse each time a voltage zero occurs on the 120 volt AC source.

4. The power supplies required for the above systems.
5. A software package containing the following modules:

a. A monitor routine to allow the user to enter and execute short programs, manipulate I/O ports, and control the operation of the system.

b. A binary floating point arithmetic package which can execute a multiplication or division in approximately one millisecond. Addition, subtraction, and conversion between floating point decimal and floating point binary numbers are also included.

c. A set of routines to allow the microcomputer system to be used as a four function calculator; the main purpose of these routines being to permit the easy testing of the above arithmetic package.

d. The software to allow the various hardware modules to be used as a DC motor speed controller. A proportional plus integral control algorithm is implemented, with keyboard alterable gain constants and RPM setpoint. Instantaneous current limiting and I^2t current limiting are included in the software, as well as routines to display parameters relevant to the operation of the control program.

Figure 1 illustrates the general block diagram of the system configured as a DC motor speed controller.

II. Digital DC Motor Control

Figure 2 illustrates the implementation of the system as a DC motor speed control device [30-38]. The digitized shaft RPM is compared to a setpoint stored in memory. This error is then used to derive a proportional control value and an integral control value using binary arithmetic. These two control values are then added; and the resulting digital value is converted to a voltage which is applied to the motor armature. The feedback path consists of a shaft mounted slotted disc [34] and the hardware and software required to convert a series of light pulses to a binary number equal to the angular velocity of the motor shaft.

The goal of the control device is to achieve constant RPM operation of the DC motor and its load. The response of the overall system can readily be optimized by the modification of the controller gain constants and the sample interval.

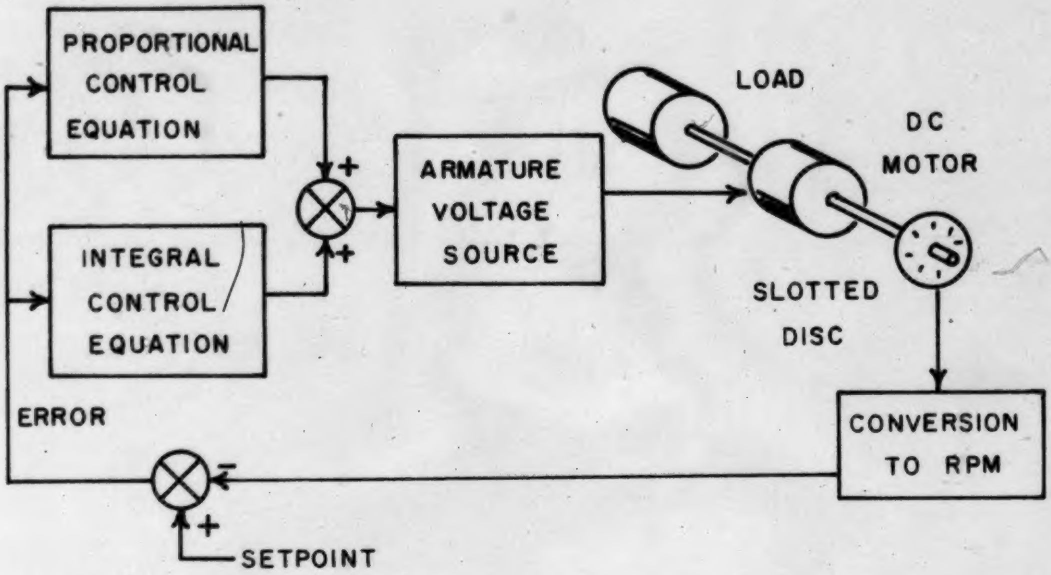

PROPORTIONAL CONTROL EQUATION
INTEGRAL CONTROL EQUATION
+
+
ARMATURE VOLTAGE SOURCE
LOAD
DC MOTOR
SLOTTED DISC
CONVERSION TO RPM
ERROR
-
+
SETPOINT

Figure 2 DC Motor Control Loop

A. The Motor Control Algorithm

The motor control is implemented using two separate digital control algorithms [39-40]: a proportional controller; and an integral controller. The sum of the outputs of the two controllers is then used to generate a motor supply voltage. The DC motor is controlled by applying this controlling voltage to the motor armature. The motor field current is maintained at a constant value, adjustable by a series resistor.

Digital Proportional Control

The proportional control output is given by

$$Cp = Kp(Rs-Ri)$$

where Cp is the proportional control output; Kp is the proportional gain constant; Rs is the RPM setpoint; and Ri is the present instantaneous RPM. The proportional gain constant and the RPM setpoint are readily modified using the system keyboard.

Digital Integral Control

The integral control output is calculated using

$$Ci = Cio + Ki(Rs-Ri)$$

where Ci is the new integral control output; Cio is the integral control output calculated at the previous sample time; and Ki is the integral gain constant. When this calculation is done in a real time system, a further variable, time, is introduced. The effective equation can then be expressed as follows:

$$Ci(t+dt) = Cio(t) + Ks(Rs-Ri)dt$$

Thus the integral gain in practice consists of the product of a gain or slope constant, Ks, which determines the rate of integration; and of the sample interval, dt. Thus

$$Ki = Ks \bullet dt$$

When the value of the sample interval is changed to determine its effect on system response, the value of Ki must thus also be altered to compensate for the changed Ks•dt product; otherwise a consistent comparison will not be made.

A digital saturating effect (this is further discussed in the next chapter) is simulated to minimize overshoot and undershoot resulting from an excessive magnitude of the value of Cio. Saturation is modeled by clipping the value of Ci at 2.0 and 0.0; which correspond to the upper (full on) and lower (full off) limits of the output voltage source.

The Control Output

The control output is given by

$$C = Cp + Ci$$

if Ki is not equal to zero, or by

$$C = Cp$$

if Ki is equal to zero. This control output, C, is converted to an average ouput voltage such that a control output of 0.0 corresponds to 0.0 volts; and an output of 2.0 corresponds to a 100% duty cycle of the full wave rectified 120 volt AC source, ie: a time average output of 108 volts.

As an example, consider the case where Kp = 0.02, Ki = 0.0 and the velocity error is 20 RPM. The resulting C is 0.02 x 20 = 0.4. This results in an average armature voltage of (0.4/2.0) x 108 = 22 volts. Average voltages have been used rather than RMS quantities, since the parameter of interest, the system's average angular velocity, is proportional to the average armature current.

B. Linearization of the Output Voltage

The actual output of the SCR circuit is a full wave rectified AC voltage, with the duty cycle of the half wave pulses program controllable. However, in order for this circuit to be useful as a controller output amplifier, there must be a linear relationship between control input and average output voltage. Since the relationship between the duty cycle and the average output voltage is nonlinear, the function used to convert the control input to a duty cycle count must compensate for this nonlinearity.

Consider one half of a cycle of a full wave rectified sine wave, as in figure 3. The SCR is turned on at time Ton, and turns off at the following current zero, Toff. The average voltage over the total cycle for a unit sinusoid is then given by

$$Vave = \frac{1}{Toff} \bullet \int_{Ton}^{Toff} \sin(wt)\,dt$$

$$Vave = \frac{\cos(w \bullet Ton) - \cos(w \bullet Toff)}{w \bullet Toff}$$

or

$$Vave = \frac{1 + \cos(w \bullet Ton)}{\pi}$$

Thus the average value ranges form 0 to $2/\pi$ depending on the turn on time, Ton. Scaling the output such that the

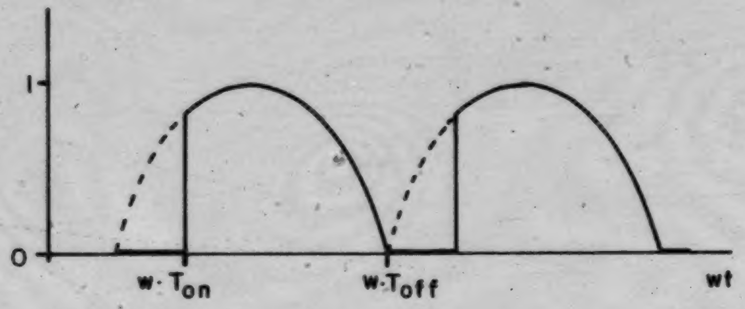


Figure 3 Variable Duty Cycle, Full Wave Rectified Sine Wave

range is from 0.0 to 2.0 (corresponding to the previously obtained control output), and solving for the turn on time in terms of C; one obtains:

$$\pi \cdot Vave = 1 + \cos(w \cdot Ton) = C$$

$$w \cdot Ton = \arccos(C-1)$$

The last equation allows the calculation of Ton; the required time delay between the leading edge of the zero crossing detector pulse and the leading edge of the SCR enabling pulse. In the DC motor speed controller, interpolation is used between the following data points:

Control Output C	Turn on Delay in Microseconds	Average Output Voltage
0.00	8333	0
0.25	6417	14
0.50	5556	27
0.75	4838	41
1.00	4167	54
1.25	3495	68
1.50	2778	81
1.75	1917	95
2.00	0	108

The given average output voltage is for a full wave rectified 120 V AC source.

C. Calculation of System RPM

The slotted disc, through its associated circuitry, provides a system interrupt every 45 degrees of shaft rotation. In order to measure system RPM, a 16 bit counter is initialized by an interrupt, then latched by the following interrupt. The down counter is started at FFFF hex, and is decremented one count by each pulse of the 1 MHz clock. The resulting count, related to the number of microseconds between pulses, is stored. The average shaft velocity over the past 45 degrees of revolution is then given by

$$\mathrm{RPM} = \frac{7.5 \times 10^6}{P}$$

where P is the number of microseconds elapsed between tachometer pulses.

D. Over Current Protection

Two types of overcurrent protection are provided; instantaneous and I^2t. After the SCR duty cycle has been updated, the armature current is read. The value thus obtained is compared to the instantaneous current limit of five times the rated steady state current of the machine; and if this value is exceeded, the SCR's are disabled and an error exit is performed.

I^2t protection is simulated by

$$Inew = Iold + Kt(A \bullet Iinst - Iold)$$

where Inew is the new 'time averaged current'; Iold is Inew from the previous sample interval, Kt is effectively a time constant; Iinst is the present instantaneous current; and A is a scaling factor. As was the case with the integral control calculation; a sample time dependence of the variable Kt is implied. Inew approaches the scaled instantaneous current in an exponential manner, with the time constant determined by Kt and the sample frequency.

It is desired that an armature current of twice the rated continuous current limit be tolerated for 10 minutes; and a current of four times the continuous limit be allowed to flow for 2.5 minutes. If Kt is set to 0.0004, and a sample interval of 100 msec is used; the value of Inew will rise to 91% of a constant A•Iinst product in 10 minutes, and to 45% of A•Iinst in 2.5 minutes; forming a reasonable approximation of an I^2t characteristic.

The scaling factor A is chosen such that with a

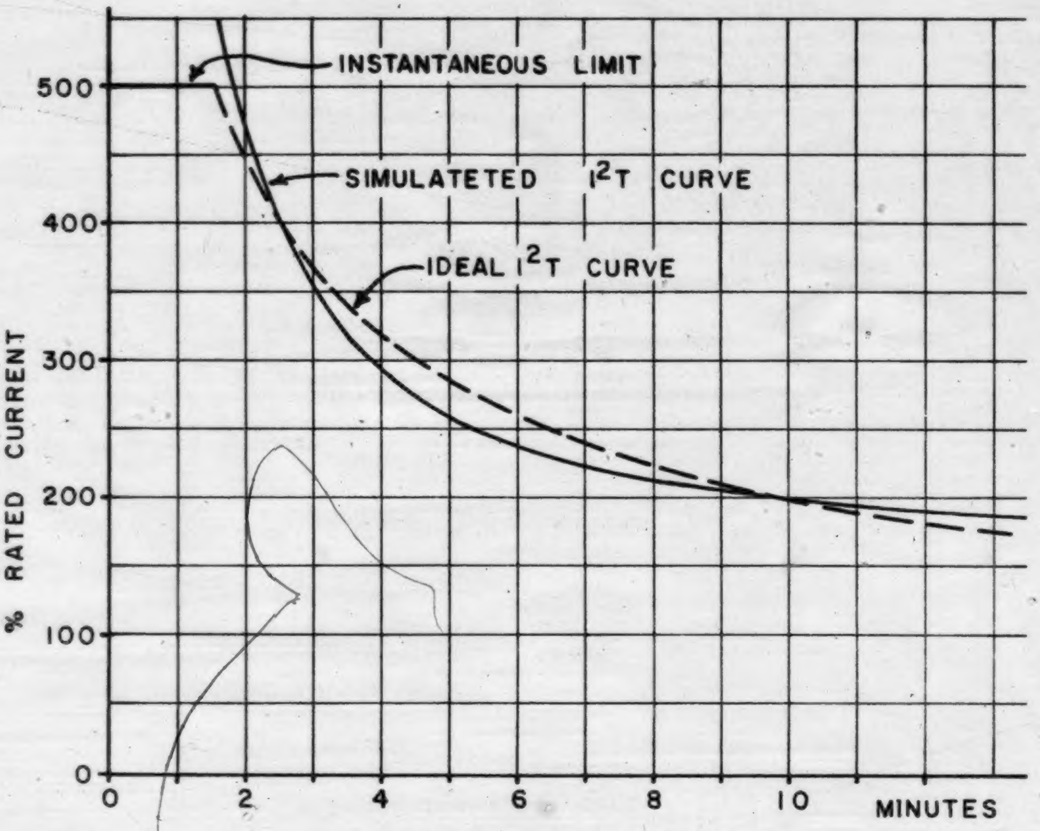


Figure 4 Continuous Current vs. Allowed Time

constant input of twice the motor's steady state armature current limit, Inew will rise from zero to a value equal to the motor's steady state current limit over a period of 10 minutes. Thus, if a current of 5.6 amps is applied with A = 0.549, at the end of 10 minutes Inew is equal to 2.8, the steady state current limit of the motor armature. At this point the SCR's will be disabled and an error message will be displayed. Figure 4 illustrates the resulting continuous current versus allowed time curve. For comparison, the ideal I^2t curve is shown in a broken line.

The magnitude of the constant Kt of 0.0004, while small in order to achieve a long time constant, is still large enough to avoid significant truncation error in the floating point arithmetic routines.

III. System Test Results

In order to evaluate system performance, an X-Y plotter was connected to a shaft mounted DC tachometer. Motor RPM was then plotted as a function of time, under the following conditions:

1. no load start from rest;
2. application of a step load while running; and
3. a step change in the RPM setpoint.

The effects of varying the sample interval were investigated, and the characteristics of saturating and nonsaturating arithmetic were compared. The effects of parameter changes on system performance were more pronounced at a nominal speed of 1000 RPM than at the motor's rated speed of 1800 RPM; for this reason the above tests were done at 1000 RPM. This exaggerated response is partly due to the lower armature back EMF at 1000 RPM. When full voltage is applied to the armature, the large difference between the applied voltage and the back EMF allows a current to flow, which is an order of magnitude greater than the steady state current. This results in a torque many times greater than what is required under steady state conditions; which will tend to emphasize ringing and overshoot. The eventual use of this system will however require the motor to run at the generator's synchronous speed of 1800 RPM; thus the system's performance as a 'governor controlled' prime mover has also been investigated.

A. DC Motor with No Control

In order to establish a frame of reference, the behavior of the DC motor was evaluated under a no control situation. The duty cycle of the SCR circuit was adjusted to give a no load rotary velocity of 1000 RPM. This effective armature voltage (Va = 28 Volts) was then held constant while load was applied to the generator. Figure 5 illustrates the resulting speed versus load characteristic. Figure 6 illustrates the corresponding no load starting response of the system with no control. The motor field current (If) for the above mentioned two graphs was held constant at 375 mA.

B. DC Motor with Proportional Control

In order to study the system's response with only proportional control, the motor was started from rest, under no load conditions, with the integral control constant set to zero. The motor field current was set to 375 mA, and the RPM setpoint (Srpm) was set to 1000 RPM. The proportional gain constant was varied over a range of 0.005 to 0.5, and graphs of RPM vs time were drawn on an X-Y plotter. A sample interval of 100 msec was used.

The resulting graphs (figures 7 to 11) illustrate the usual characteristics of a proportional control system. A system with a fairly low proportional gain constant (Kp), in

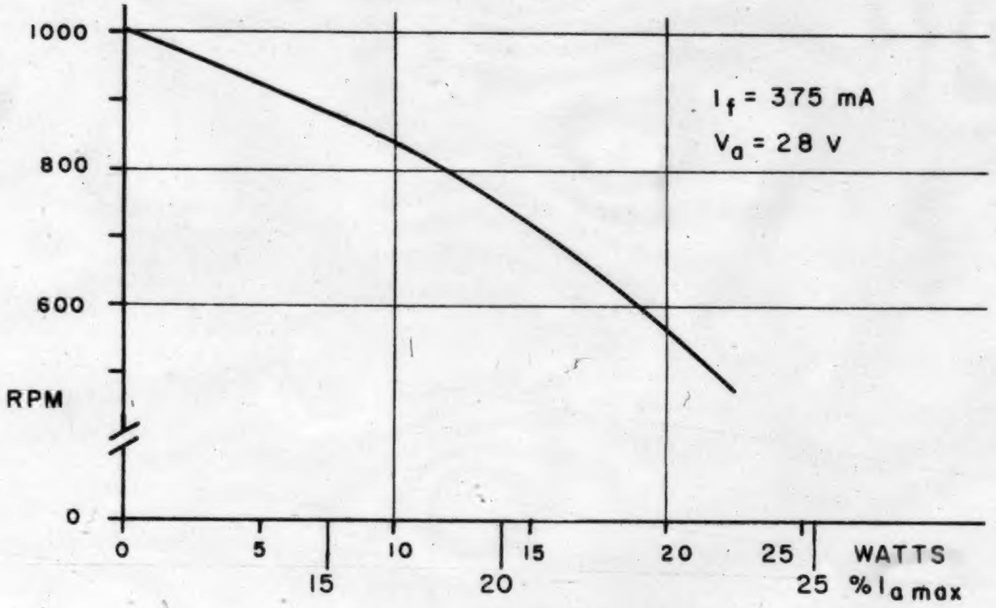


Figure 5 Speed Load Characteristic with No Control

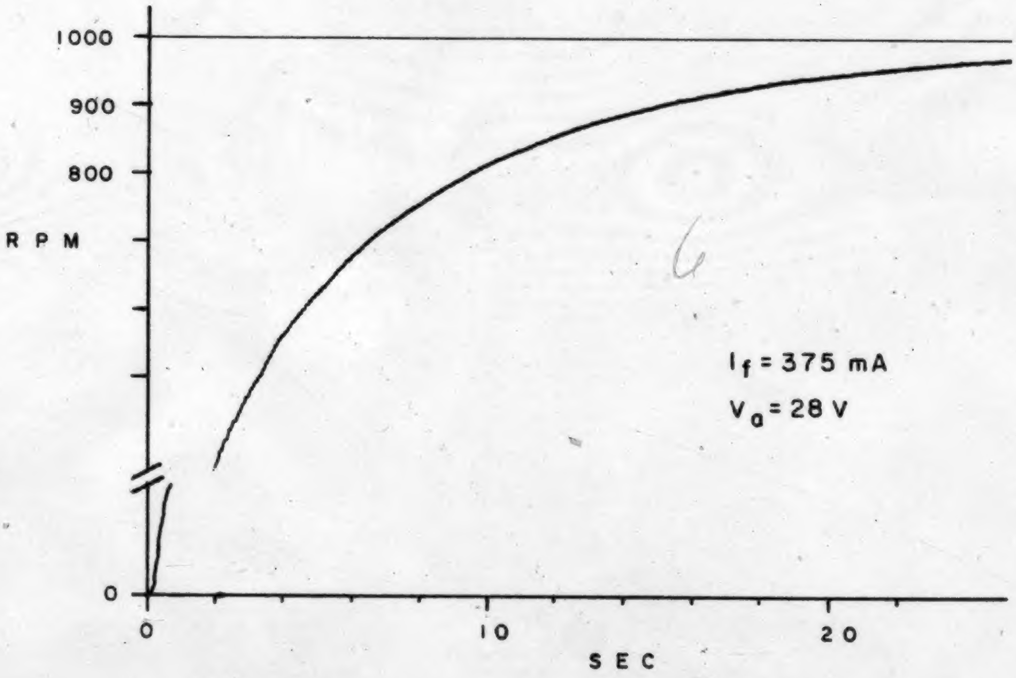


Figure 6 No Load Start; No Control

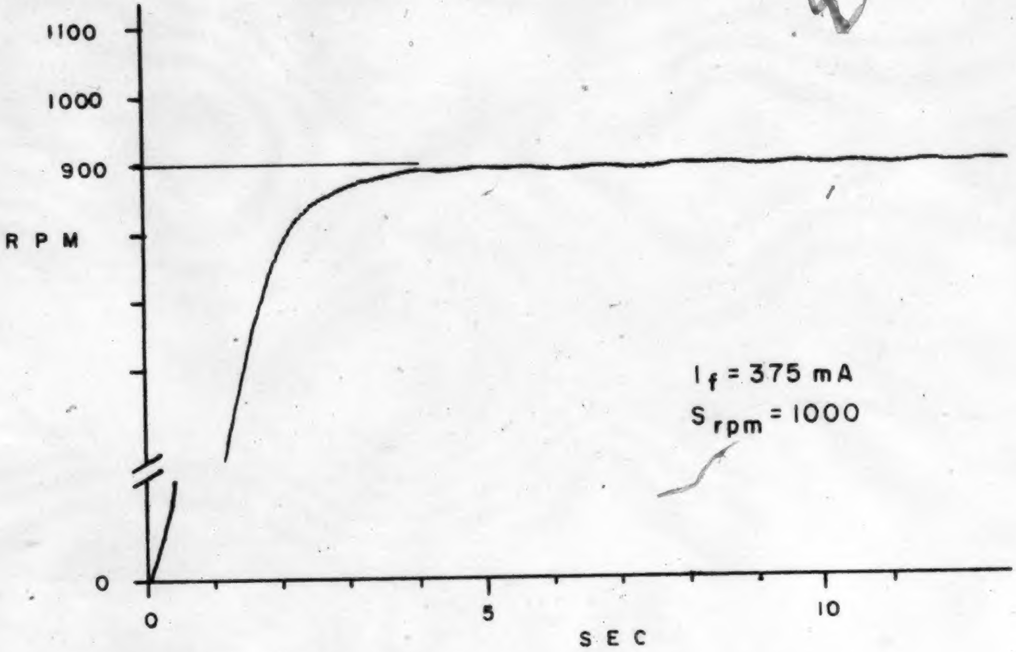


Figure 7 No Load Start; Kp = 0.005

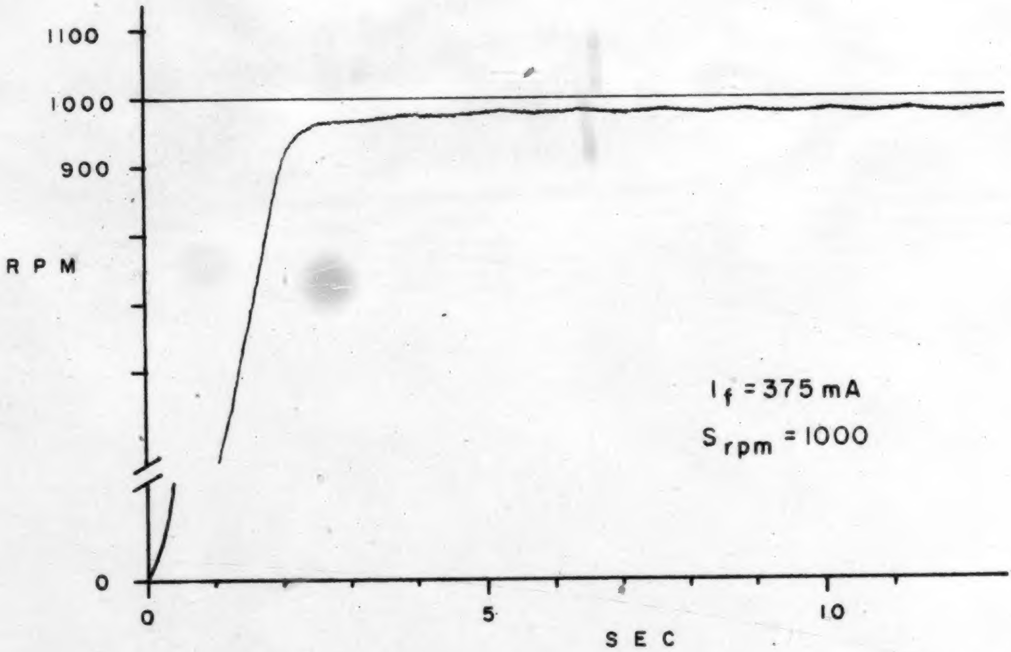


Figure 8 No Load Start; Kp = 0.02

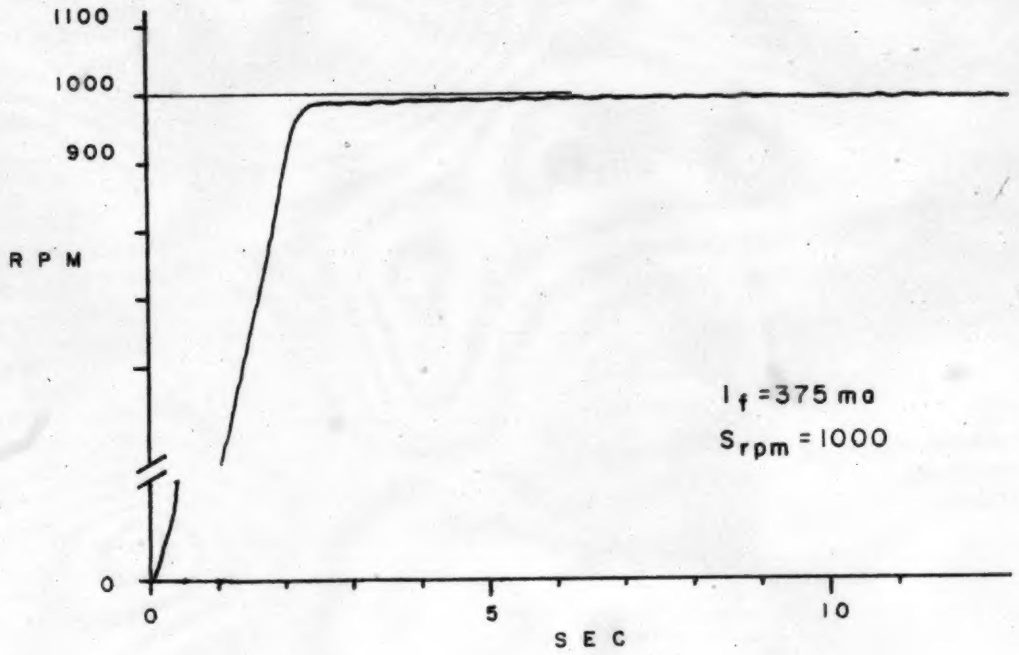


Figure 9 No Load Start; Kp = 0.05

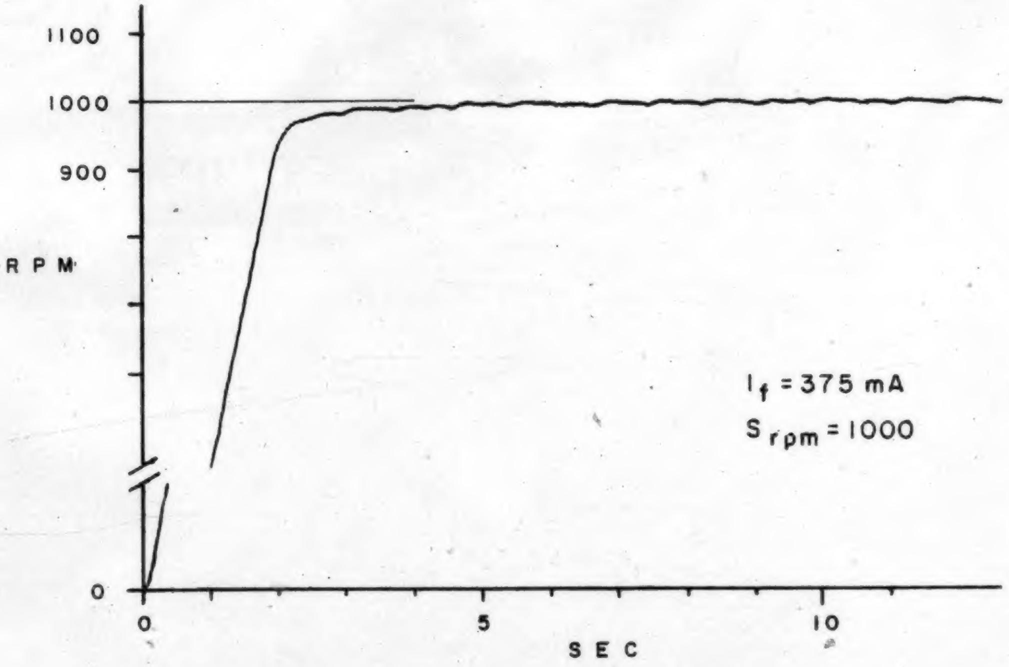


Figure 10 No Load Start; Kp = 0.1

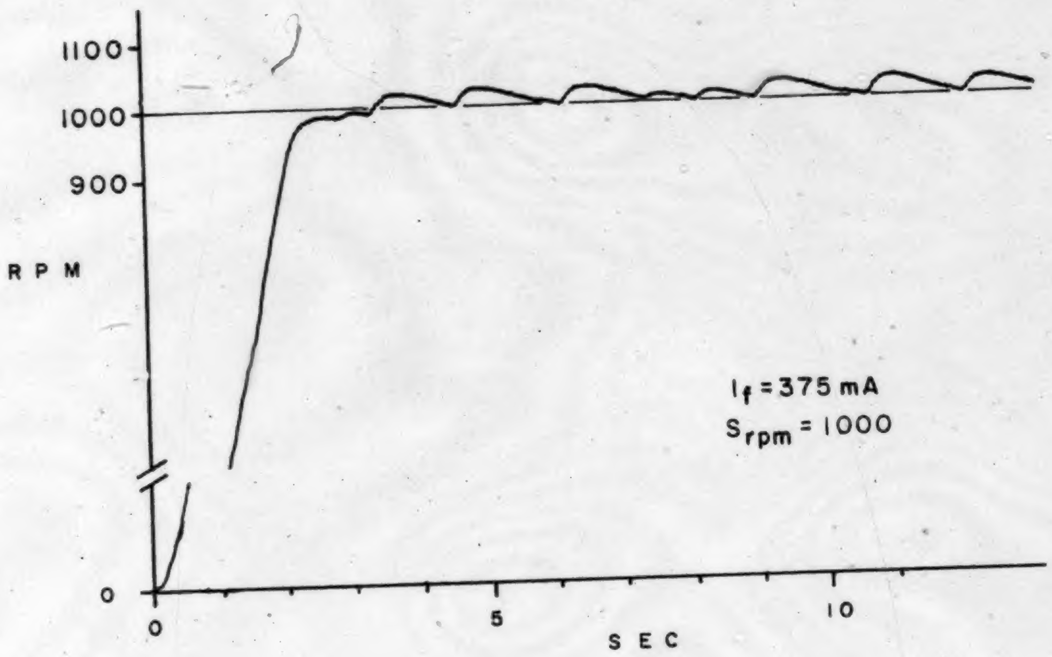


Figure 11 No Load Start; Kp = 0.5

this case 0.005 or less, exhibits fairly sluggish response (figure 7), as well as an appreciable steady state error, to a step input. An increase in proportional gain results in a sharper corner as the motor reaches 1000 RPM, along with a significant decrease in steady state error. When the proportional gain reaches 0.1 (figure 10) a different situation arises. Instead of a noticable steady state error, there now is an oscillation about the set point. Due to the high gain, the compensation for a small error results in the system overshooting in the opposite direction. This becomes quite extreme with a gain of 0.5.

Figure 11 also illustrates the difference in the system time constants during acceleration and deceleration. With

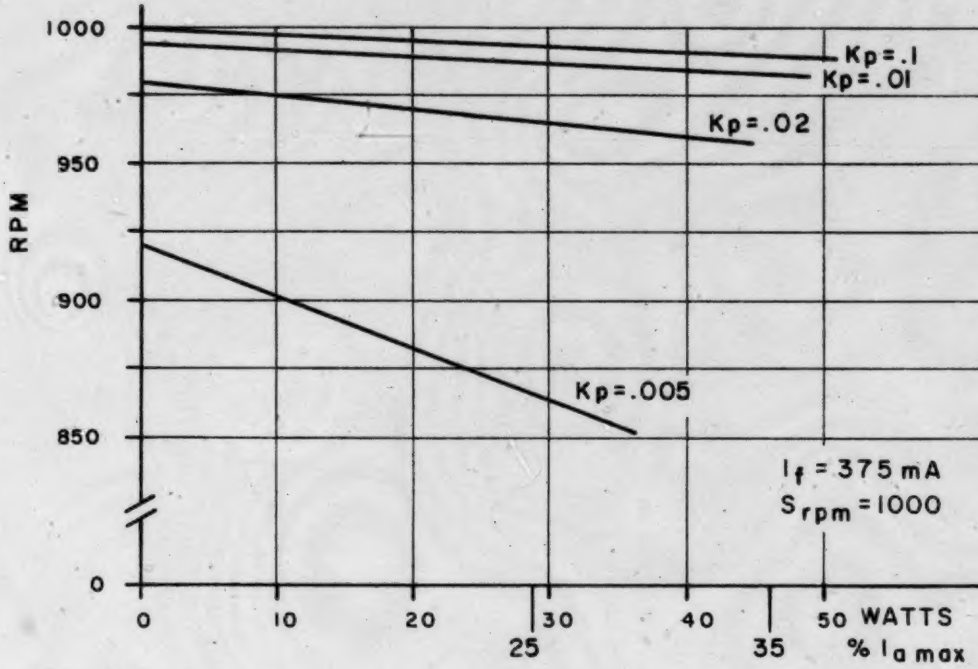


Figure 12 Speed Load Characteristic at 1000 RPM

full voltage applied to the armature, the system will accelerate at a rate of 500 RPM per second, while under no load conditions it will coast to a stop at only -50 RPM per second. Since this is a discrete data control system, the large difference between the two time constants results in the error being rather one sided. That is, with a large gain constant, the system is capable of significant over acceleration between data samples; while it takes several sample intervals for significant deceleration to take place. Decreasing the sample interval would of course reduce the error in both directions.

There are also, visible in the figures, apparent high frequency oscillations, of less than one millimeter peak to peak amplitude (corresponding to less than 10 RPM peak to peak), superimposed upon the RPM versus time graphs. These oscillations are not actually present in the motor's angular velocity, but are a characteristic of the recording system used to produce the RPM versus time curves. These oscillations, at a frequency of approximately 17 Hz at 1000 RPM, are the result of the DC tachometer's commutated output. A moderate amount of filtering has been used to reduce this noise to an acceptable level, but it could not be completely eliminated without affecting the X-Y plotter's ability to accurately track the system's response.

Speed Load Characteristics at 1000 RPM

The steady state error of a proportional control system increases as the system load is increased. Figure 12 illustrates the speed load characteristic for several values of Kp, with a velocity set point of 1000 RPM.

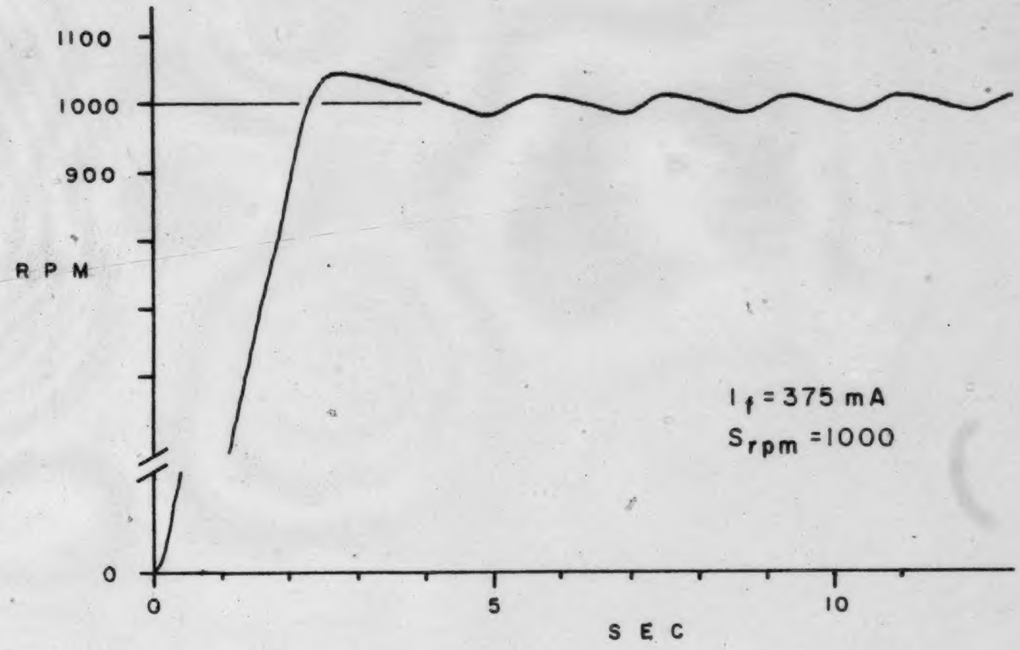


Figure 13 No Load Start; Kp = 0.002, Ki = 0.01

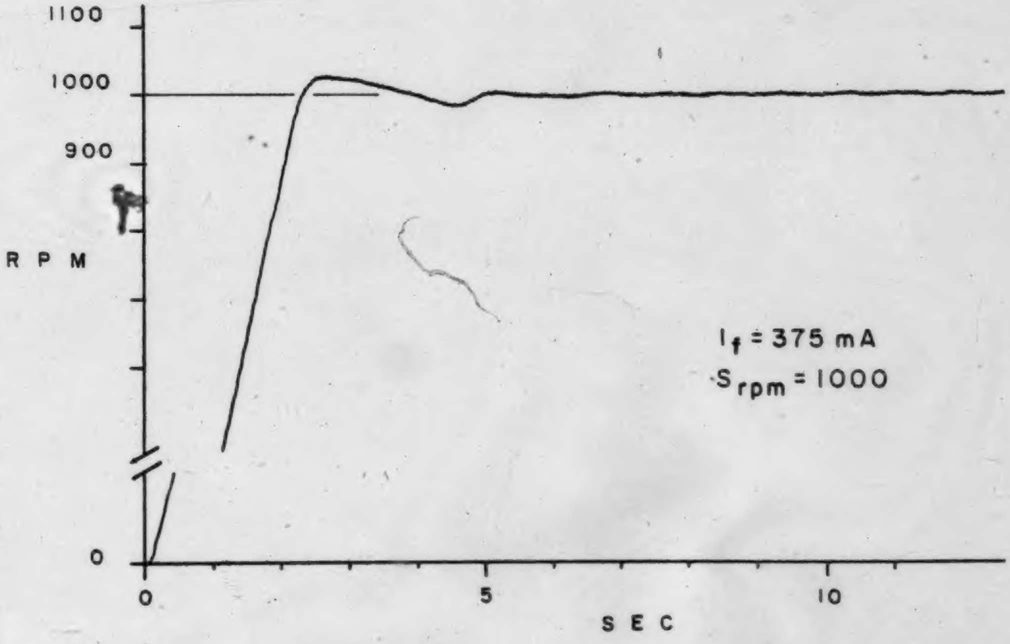


Figure 14 No Load Start; Kp = 0.01, Ki = 0.01

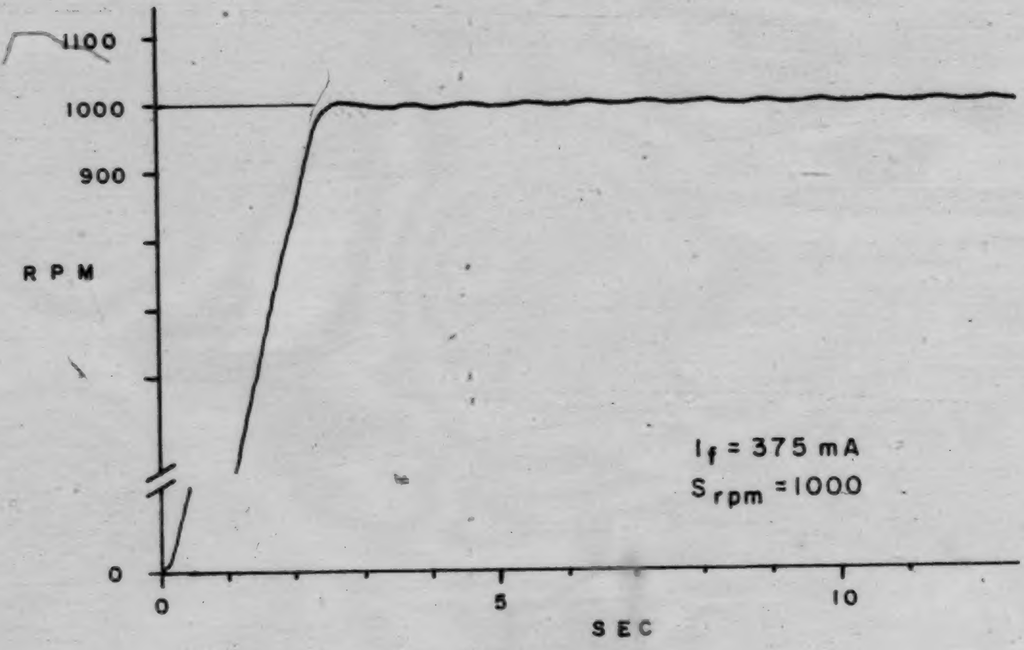


Figure 15 No Load Start; Kp = 0.05, Ki = 0.01

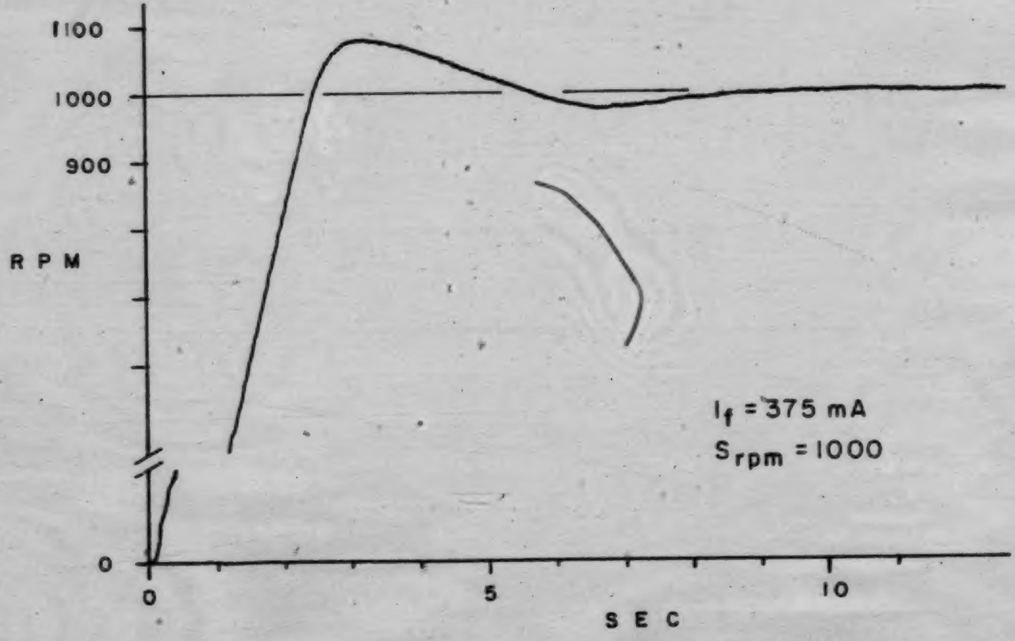


Figure 16 No Load Start; Kp = 0.01, Ki = 0.001

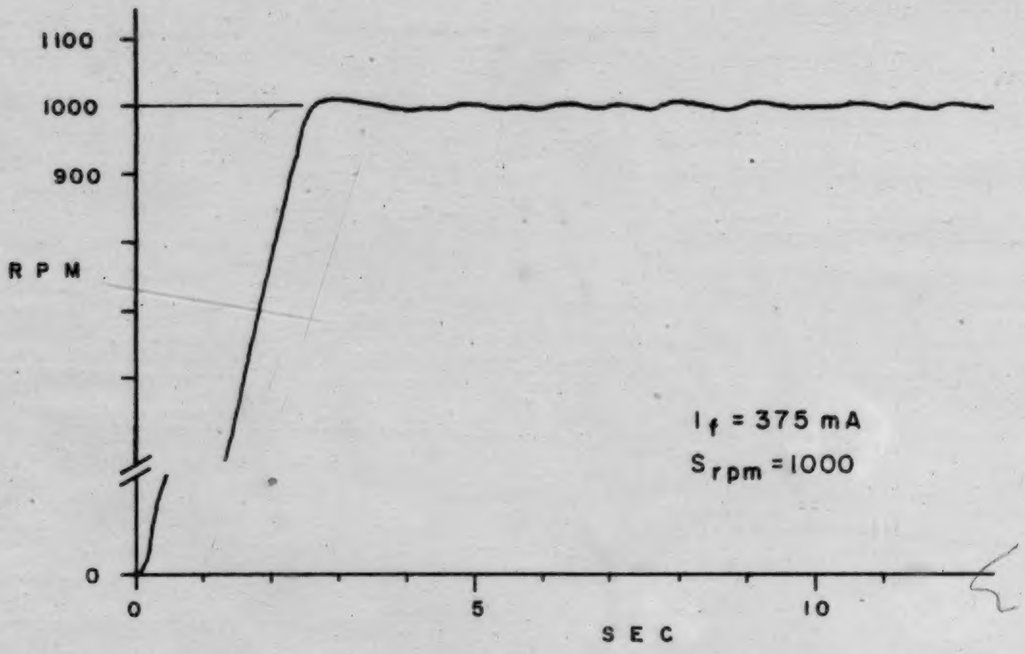


Figure 17 No Load Start; Kp = 0.01, Ki = 0.04

C. Proportional Plus Integral Control

System response was next evaluated with proportional plus integral control. As expected, the addition of integral control brought the average error in the steady state response to a step input to zero. In PI control, the proportional gain constant has a direct bearing on how fast the oscillations of the integral control output are damped out. Figure 13 illustrates lightly damped ringing with a proportional gain of 0.002. Increasing the proprtional gain while holding the integral gain at a constant value, results in a significant reduction in ringing; as well as a decrease in overshoot. A further increase in Kp to 0.05 (figure 15) results in almost complete elimination of overshoot and ringing; at the expense of introducing oscillations as a result of too much proportional gain (as illustrated in figure 10).

The effects of the integral gain constant (Ki) were examined by holding the proportional gain constant fixed, while Ki was varied. Figure 16 illustrates the increased overshoot as the integral gain is reduced; while figure 17 illustrates the significantly reduced overshoot and settling time obtainable with higher integral gain. On the other hand, the latter exhibits significant steady state oscillations due to the high integral gain; whereas in figure 16 these oscillations are insignificant. This suggests a tradeoff between rapid damping of overshoot or ringing, and the minimization of steady state oscillation. For example; values of Kp = 0.01 and Ki = 0.01 result in acceptable system response.

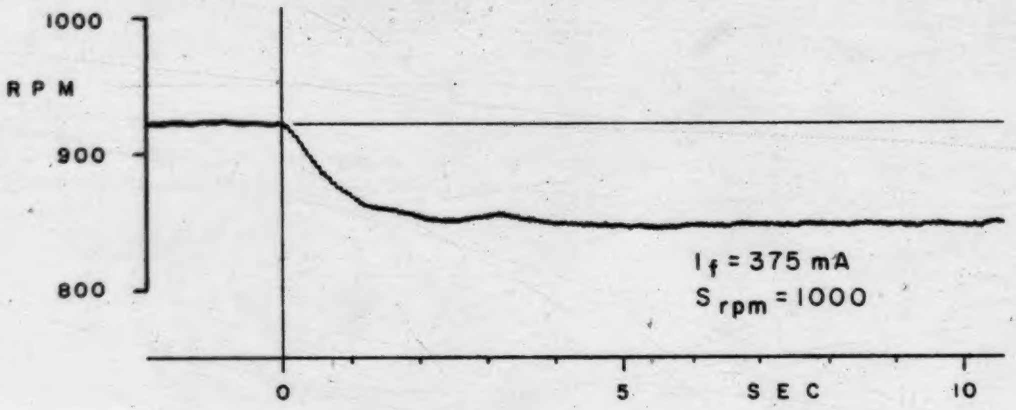


Figure 18 40 Watt Load Step; Kp = 0.005

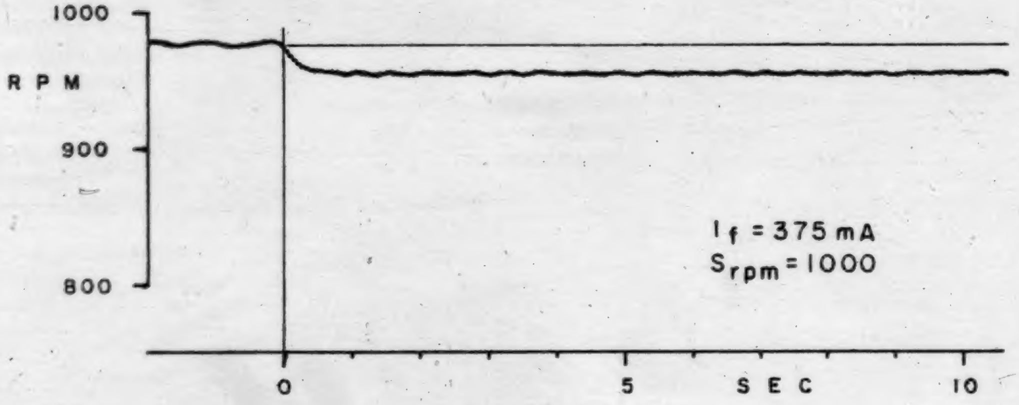


Figure 19 40 Watt Load Step; Kp = 0.02

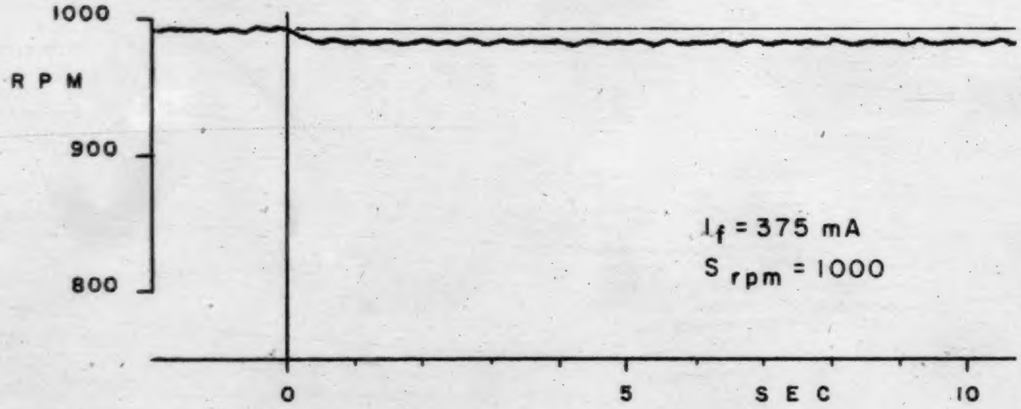


Figure 20 40 Watt Load Step; Kp = 0.05

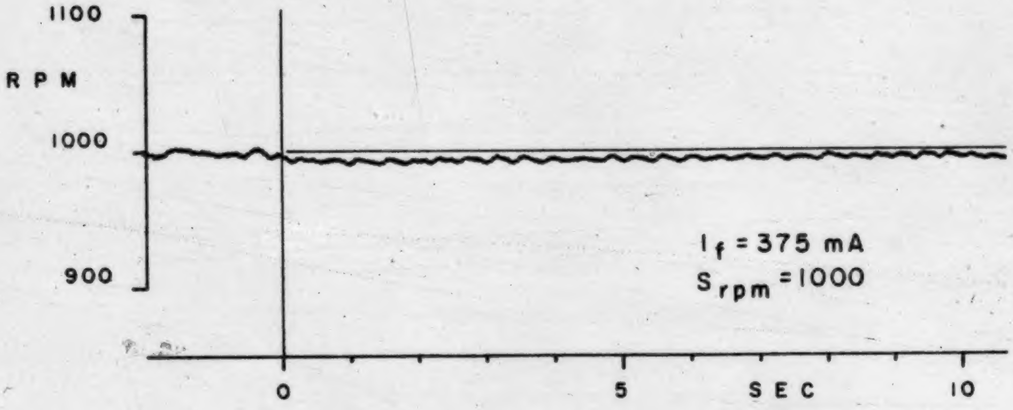


Figure 21 40 Watt Load Step; Kp = 0.1

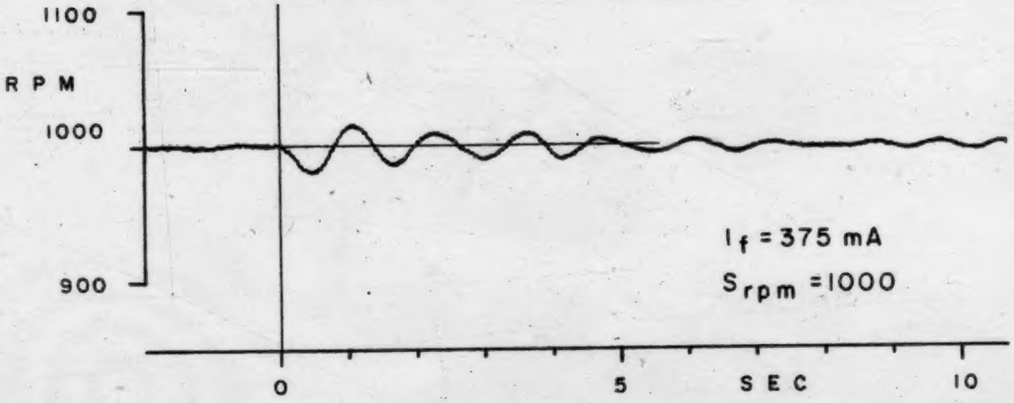


Figure 22 40 Watt Load Step; Kp = 0.002, Ki = 0.01

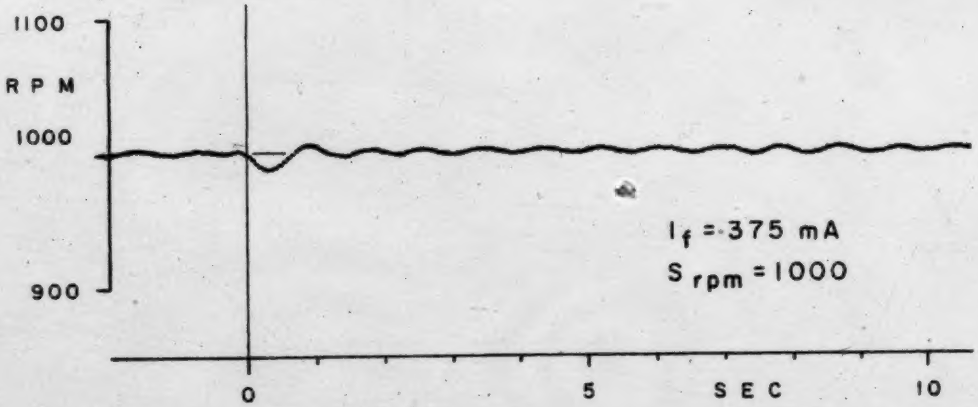


Figure 23 40 Watt Load Step; Kp = 0.01, Ki = 0.01

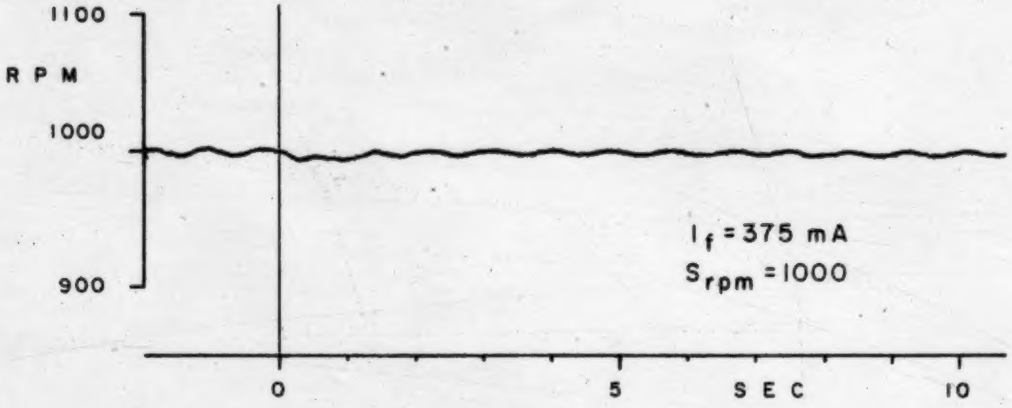


Figure 24 40 Watt Load Step; Kp = 0.05, Ki = 0.01

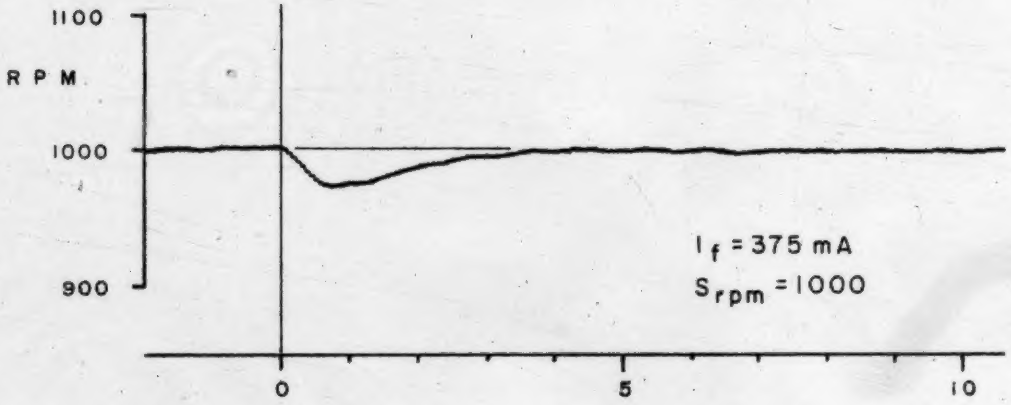


Figure 25 40 Watt Load Step; Kp = 0.01, Ki = 0.001

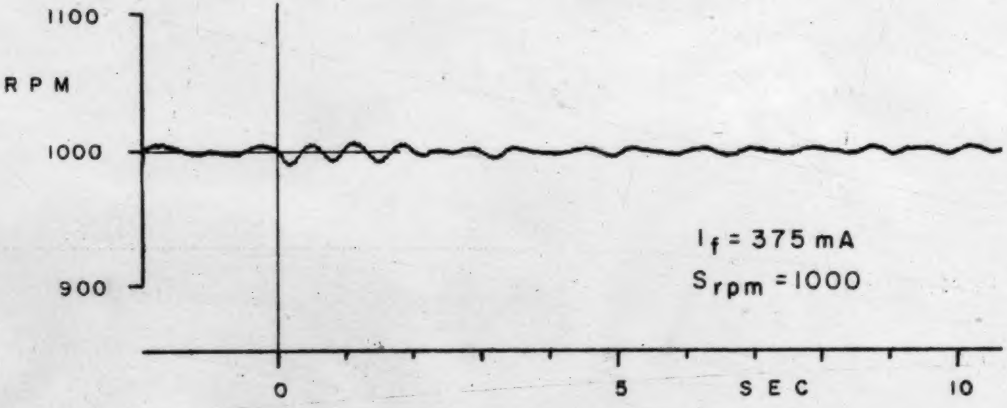


Figure 26 40 Watt Load Step; Kp = 0.01, Ki = 0.04

D. The Application of a Step Load

Once the no load steady state velocity had been reached, a step load was placed on the system by switching the generator exciter field current on. The 730 mA exciter current resulted in 40 watts of power being dissipated in the generator's resistive load. The motor shunt field current remained at the previous 375 mA.

The resulting RPM vs time graphs (figures 18 to 26) exhibit a behavior similar to that observed when the system was started from rest. Where only proportional control is used, the velocity decrease as load is applied is more pronounced for small values of Kp. As Kp is increased, this average, loaded velocity error decreases; at the expense of increased steady state oscillations, as before.

The addition of integral control again brought the steady state velccity back to the set point. The effect of Kp on the damping of ringing and overshoot is again apparent. Figure 25 again illustrates the slower response with reduced proportional gain; while figure 26 typifies the oscillations accompanying the faster response resulting from an increased Ki.

E. Changes in the RPM Setpoint

The previous nine graphs were repeated once more. However, instead of the application of a step load, the RPM set point was changed from 1000 to 1100 RPM. Due to its dependence on RPM; the sample interval decreased to 90 msec at 1100 RPM, while it remained at 100 msec at 1000 RPM. The motor field current remained at 375 mA while the generator exciter remained open circuited. The RPM setpoint (Srpm) was changed from 1000 RPM to 1100 RPM, using the keyboard monitor, while the motor was running.

F. The Effects of Changing the Sample Interval

Due to the time required for the calculation of the various control parameters, the minimum possible sample interval is in the order of 10 msec. By recording the system velocity under no load starting conditions, it was determined that there was little performance improvement to be gained by reducing the sample interval to 50 msec or less. Also, increasing the sample interval to 200 msec (see figure 36) did not significantly deteriorate system performance (as compared to figure 14).

A further increase in sample time will however result in unstable operation. Starting the system from rest with Kp = 0.01, Ki = 0.025, and a sample interval of 250 msec will result in unstable operation. Under no load starting conditions, the system RPM will increase to approximately 1200 RPM; where it will remain without returning to its

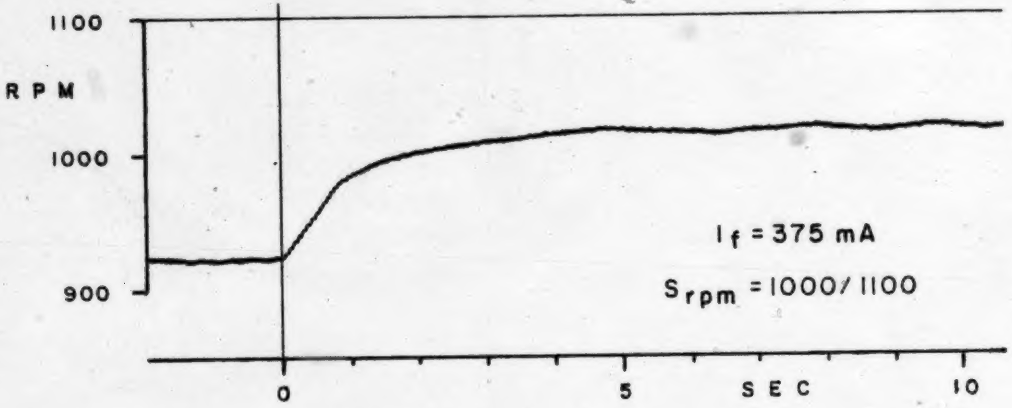


Figure 27 RPM Setpoint Change; Kp = 0.005

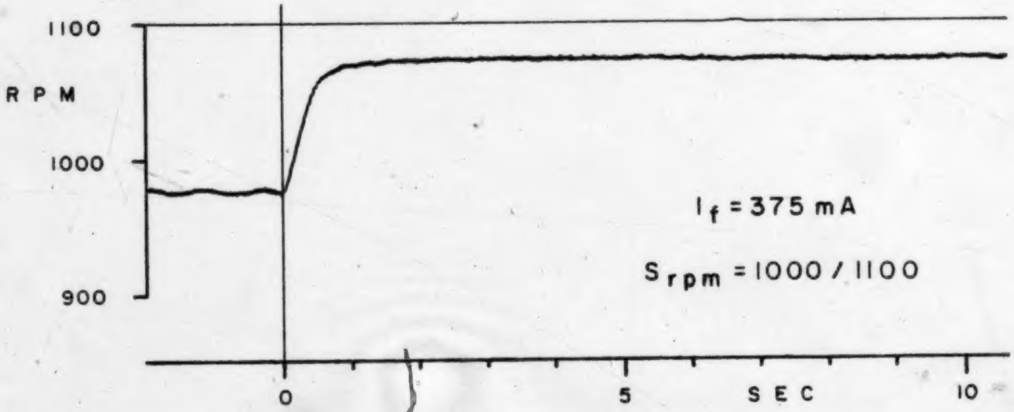


Figure 28 RPM Setpoint Change; Kp = 0.02

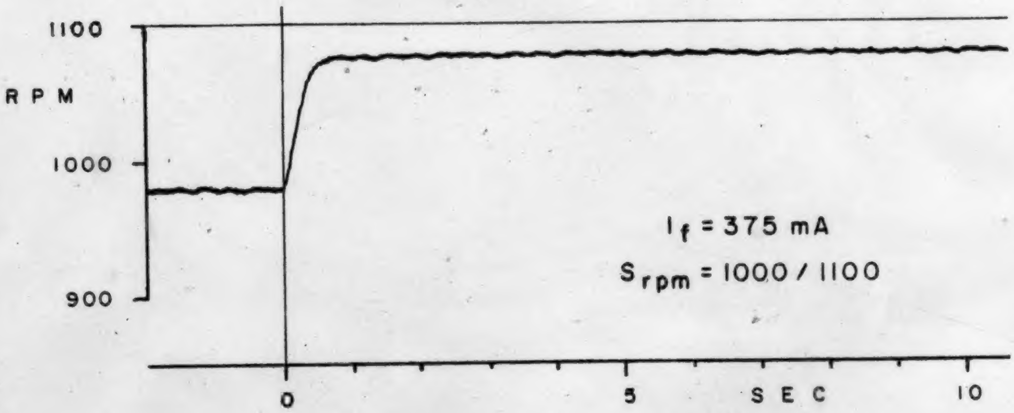


Figure 29 RPM Setpoint Change; Kp = 0.05

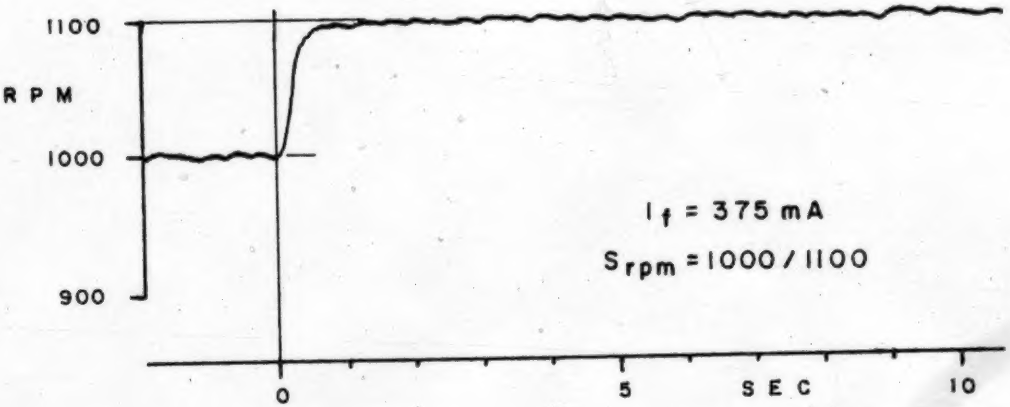


Figure 30 RPM Setpoint Change; Kp = 0.1

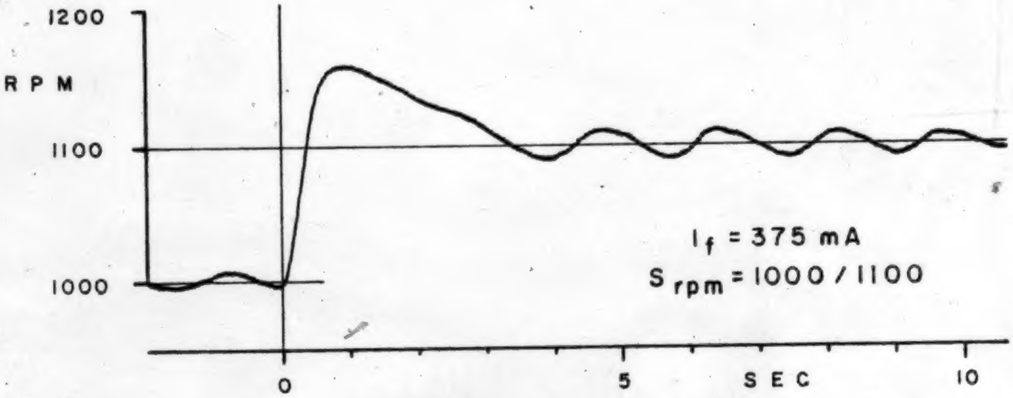


Figure 31 RPM Setpoint Change; Kp = 0.002, Ki = 0.01

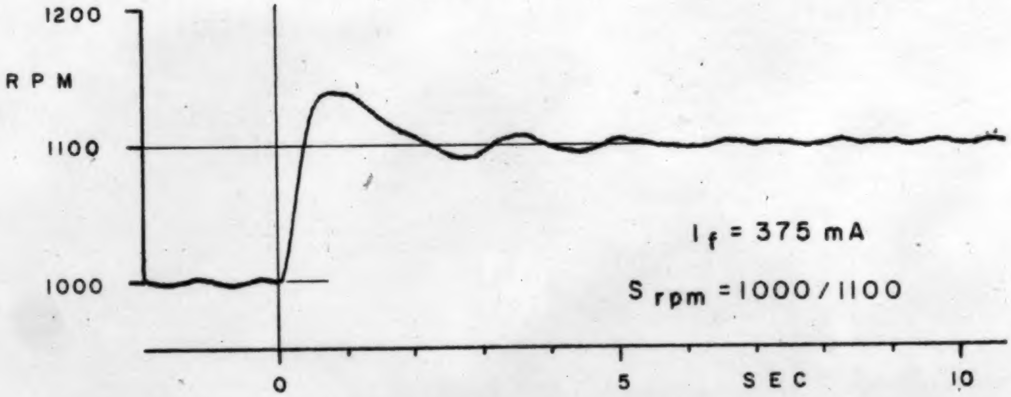


Figure 32 RPM Setpoint Change; Kp = 0.01, Ki = 0.01

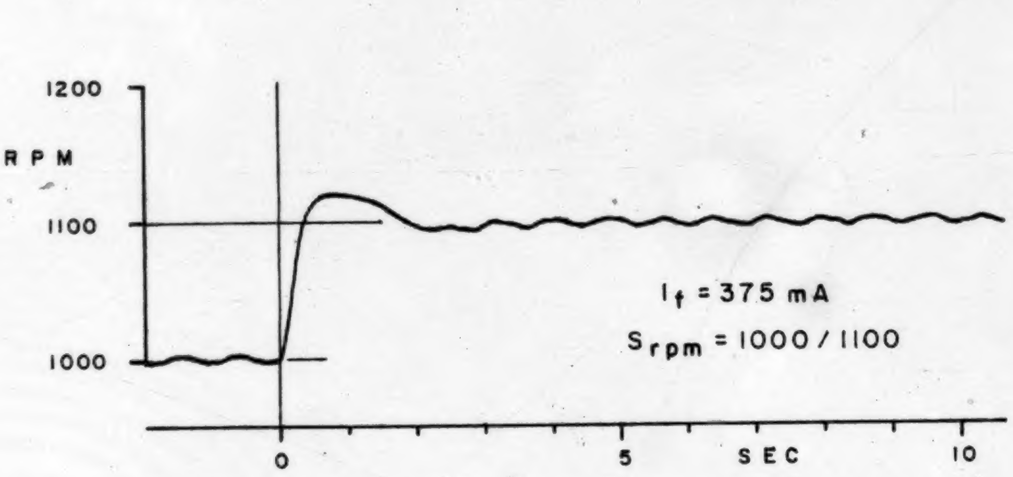


Figure 33 RPM Setpoint Change; Kp = 0.05, Ki = 0.01

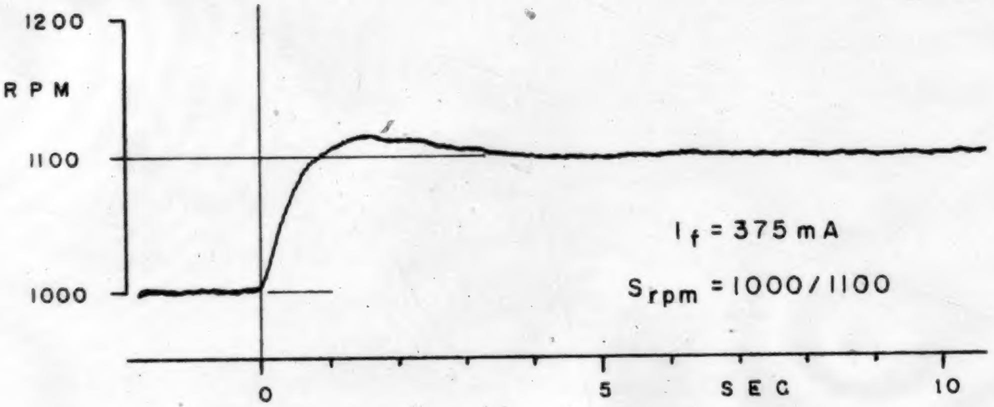


Figure 34 RPM Setpoint Change; Kp = 0.01, Ki = 0.001

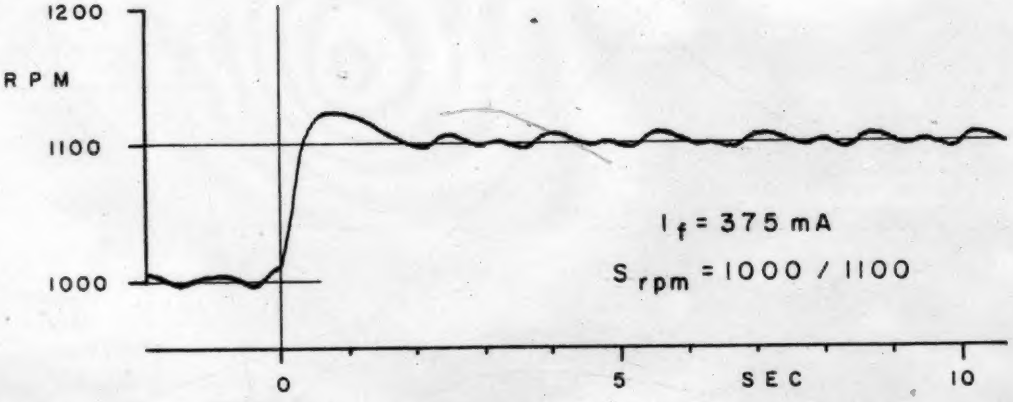


Figure 35 RPM Setpoint Change; Kp = 0.01, Ki = 0.04

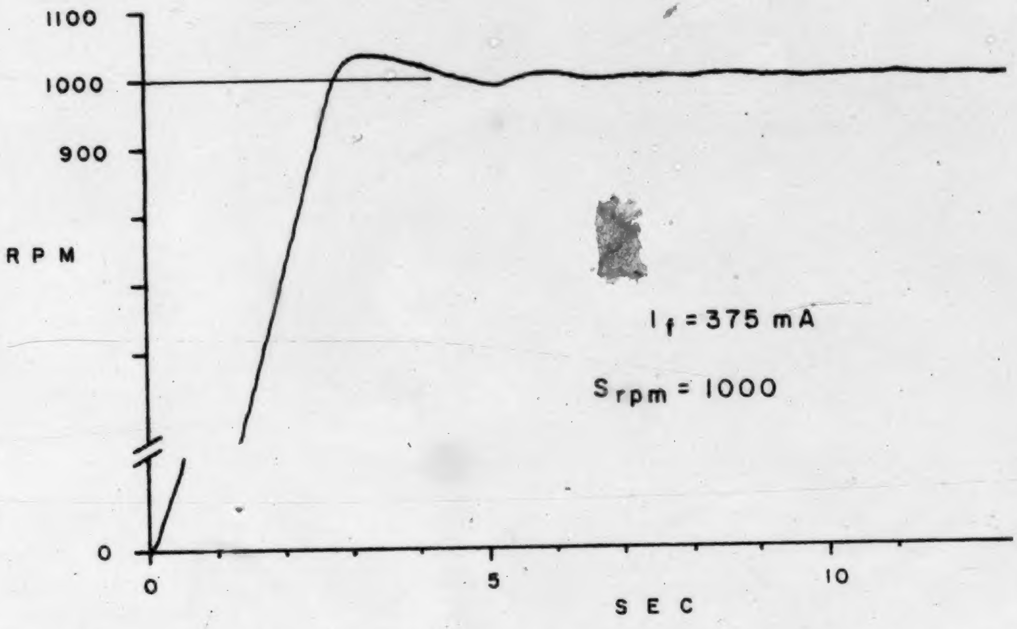


Figure 36 200 msec sample interval, Kp = 0.01, Ki = 0.02

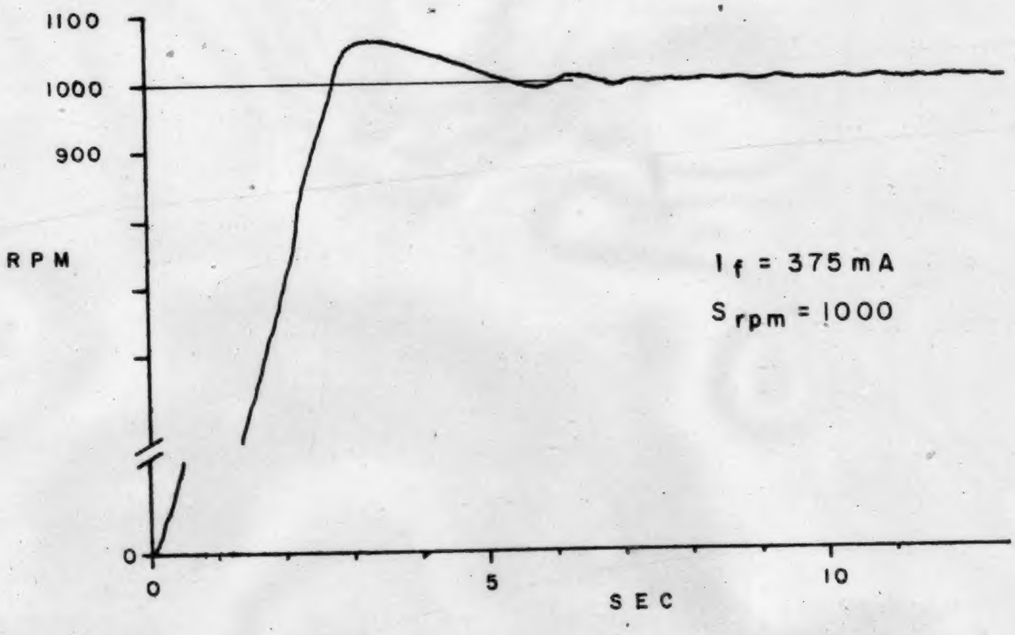


Figure 37 250 msec sample interval, Kp = 0.02, Ki = 0.025

setpoint. If however, a 40 watt load is applied, the system will return to 1000 RPM, maintaining operation there with moderate oscillations, even after the load is removed. Starting the system under a light load will also result in stable operation at 1000 RPM. In figure 37, a stable start from rest has been achieved by increasing Kp to 0.02; thus increasing the effective damping on the system

The integral gain, Ki, contains a sample interval term, dt, as explained previously. Thus the value of Ki must be adjusted when the sample interval is changed in order to maintain the same overall integral gain. In the above mentioned tests, Ki was adjusted such that the effective integral gain was equal to that obtained with Ki = 0.01 and a sample interval of 100 msec.

G. Saturating and Non-Saturating Arithmetic

In an analog integral control system, the range of variables is limited by the saturation voltage of the components involved. In a digital system, this limitation is only imposed by an overflow error in the numbers involved. In a practical system, using floating point arithmetic, this truncation does not occur until the numbers involved exceed their practical values by several orders of magnitude. The effect of this can best be visualized as follows. Consider starting the motor generator set from a standstill with integral control playing a part of the overall control scheme. Initially the error is great, and

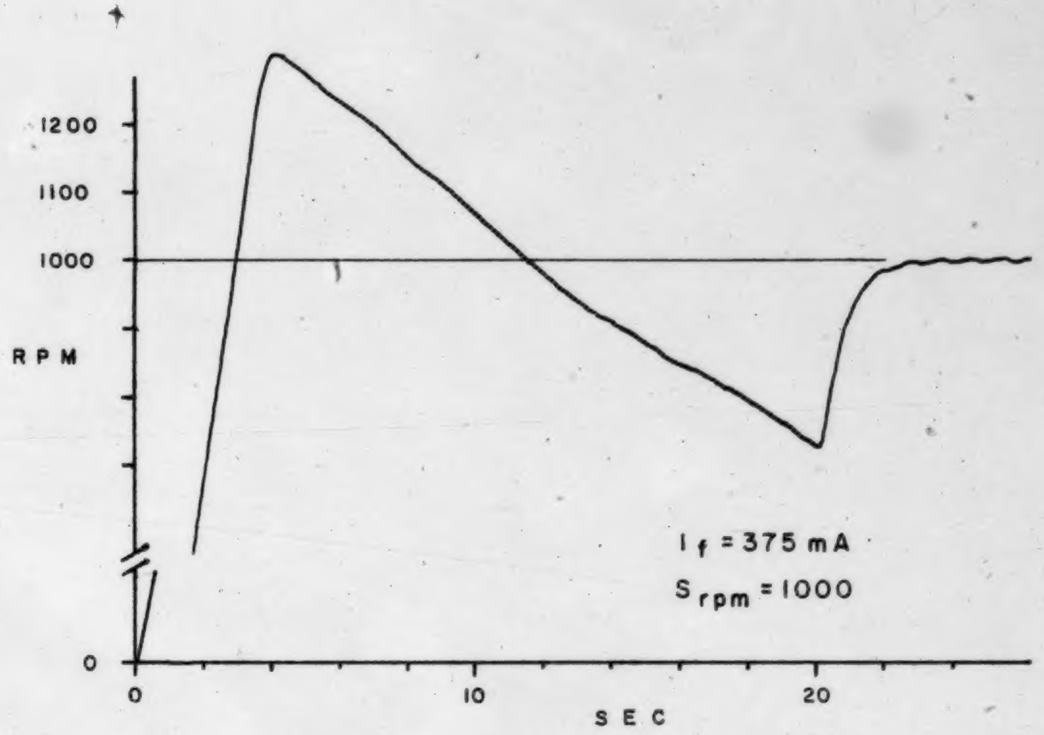


Figure 38 Nonsaturating Arithmetic; Kp = 0.05, Ki = 0.01

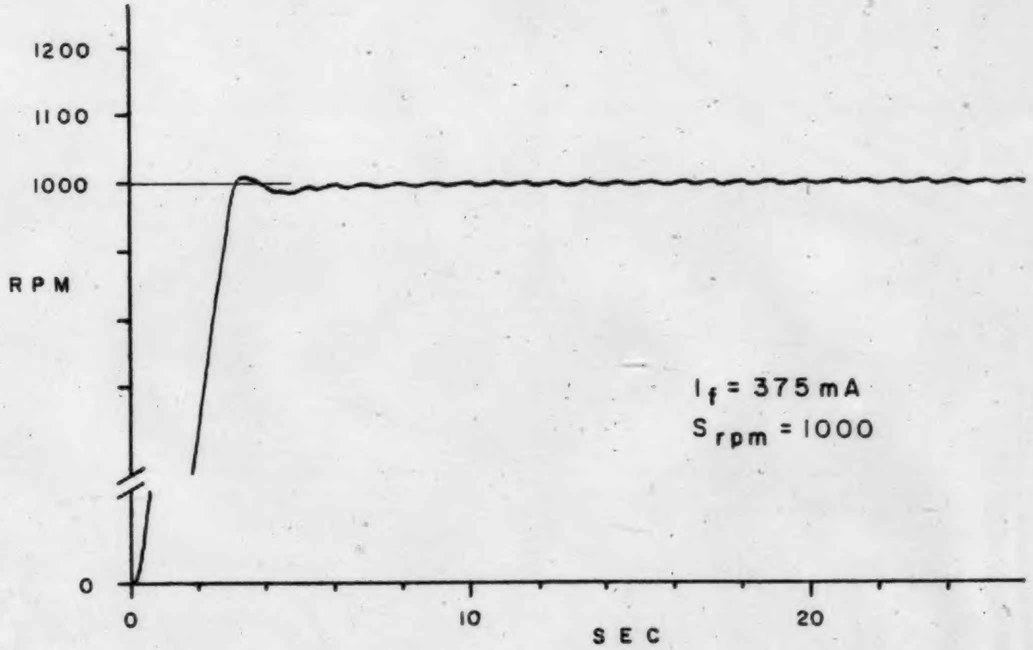


Figure 39 Saturating Arithmetic; Kp = 0.05, Ki = 0.01

so is the resulting integral control output. Thus the control output is driven to the upper limit, and the corresponding integral control value is stored. At the second sample interval, the output is still driven to its upper limit, and another large integral control value is added to the previous one. This process repeats every sample interval, and by the time the system velocity reaches 1000 RPM, the stored integral control value is of considerable magnitude; much larger than the ceiling value of the output stage. Thus, even after the system RPM has overshot the setpoint and the error is now in the positive direction, considerable time is required to 'integrate' the integral control value down to zero, resulting in a very large overshoot. Eventually, the stored integral control value becomes small enough, and the system begins to ramp down to 1000 RPM again. However, considerable undershoot results, and the cycle repeats until the system eventually achieves normal operation. Figure 38 illustrates a typical response graph for a system using non saturating arithmetic.

Fortunately, it is not necessary to let an integral control output reach a value in excess of the corresponding limits imposed by the output circuitry. Thus in this system, if the integral control output exceeds the limits of the output circuit (0.0 and 2.0), the integral control value is 'clipped' at the proper value before it is stored for use at the next sample interval. This results in no change in performance at steady state or for small transients, since under these conditions the limits are never reached. However, during large transients, as in a system start up, performance is dramatically improved, as illustrated in figure 39. All parameters in the two above mentioned graphs are identical, the only difference being the fact that the 'saturating' program was bypassed in the former.

H. Operation at 1800 RPM

Figure 40 illustrates a typical no load system startup with the setpoint at 1800 RPM. Here the sample interval counter is set such that a 100 msec sample interval is obtained at 1800 RPM. The motor field current is set to 250 mA. Under these operating conditions oscillations about the setpoint are reduced, even with higher values of Ki and Kp. This is due to the effect of the higher RPM on the armature back EMF and on the frictional and windage load. Since the armature back EMF is significantly higher at 1800 RPM than at 1000 RPM, the maximum armature current is reduced. As a result, the torque correspnding to maximum applied armature voltage is also reduced. Thus the amount of over acceleration between sample intervals is reduced, resulting in slightly reduced error. The higher RPM also results in a significant increase in frictional load, as well as an increased load due to motor and generator windage. These effects increase the rate at which the system will coast to a lower RPM, and thus reduce the deceleration time constant to a more reasonable value.

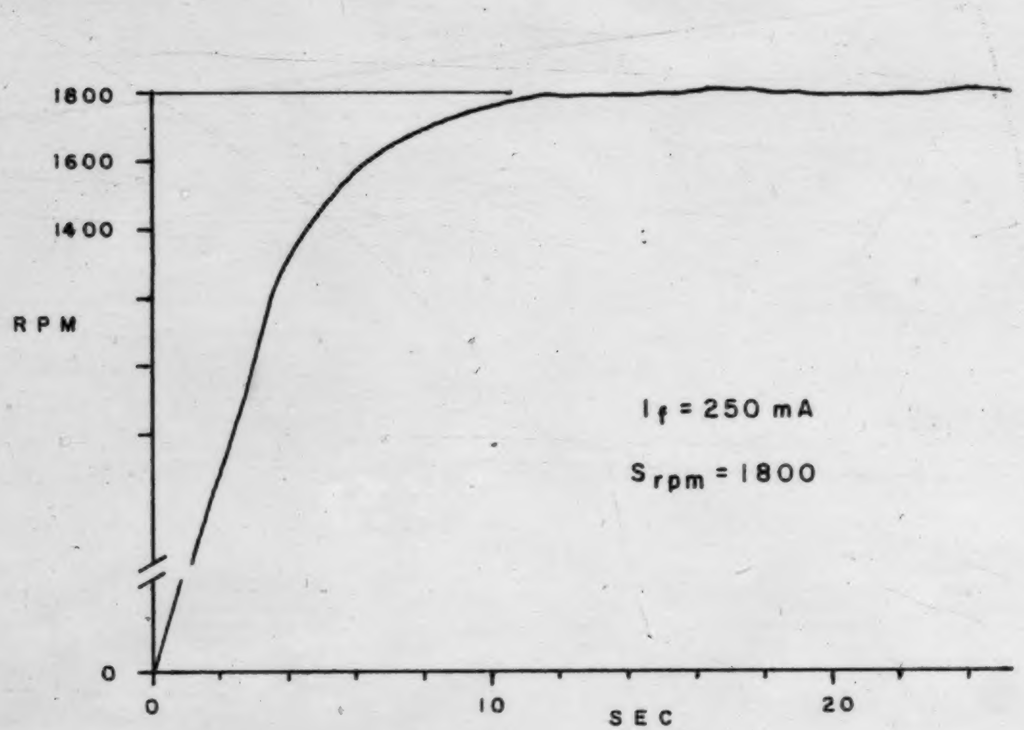


Figure 40 No Load Start at 1800 RPM; Kp = 0.05, Ki = 0.02

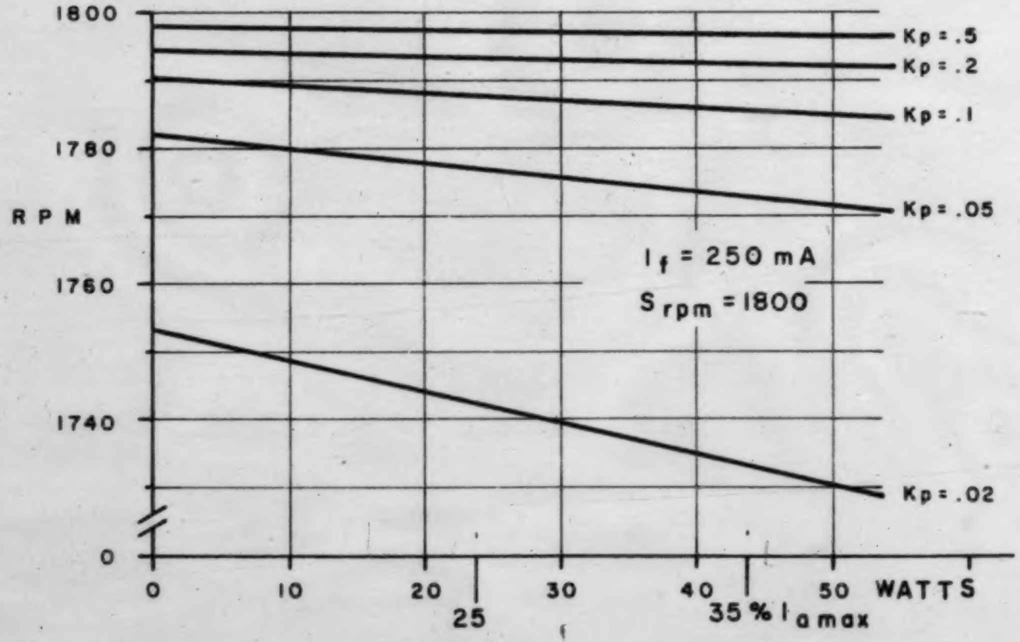


Figure 41 Speed Load Characteristic at 1800 RPM

Speed Load Characteristic at 1800 RPM

Figure 41 portrays the system speed load characteristic at 1800 RPM. Here the motor shunt field current was set to 250 mA, and a generator load consisting of Y connected 500 ohm resistors was connected. The generator load was varied by changing the generator excitation current. The graph as drawn indicates straight speed-load lines; in practice they are straight only to about 60 watts. At this point there is a rolloff, due to the limitations imposed by the source voltage and current driving the motor.

I. Steady State Error with PI Control

With a reasonable choice of constants, the average steady state system velocity error can be reduced to practically zero. However, there will always be an instantaneous error due to the slight oscillations about the set point. As previously mentioned, the magnitude of these oscillations can become quite large if excessively large values of Ki and Kp are used. Figure 42 indicates typical error values obtained with a 100 msec sample interval.

It is readily apparent that the magnitude of this error can be reduced by taking the following steps:

1. reduction of Ki and Kp to reasonable values;
2. increasing the sample rate;
3. increasing the system velocity, as explained above; and

Kp	Ki	RPM setpoint	load	RMS error
.01	.01	1000	-	.4 %
.01	.01	1000	40 W	.35 %
.01	.001	1000	-	.25 %
.01	.04	1000	-	.65 %
.05	.02	1800	-	.4 %
.05	.02	1800	50 w	.07 %

Figure 42 RMS Velocity Error Under Typical Operating Conditions

4. increasing the load on the system, thus reducing the deceleration time constant.

IV. Keyboard Operation of the Microcomputer System

In this section, the keyboard operation of the computer system is discussed from a users point of view. The keyboard supporting software consists of two main components; a system monitor and an arithmetic package. The system monitor allows the user to interrogate memory and I/O ports, write to alterable memory locations or I/O ports, and initiate program execution at any desired address. Note that in the monitor system, hexadecimal notation is used exclusively. The arithmetic package consists of a series of floating point arithmetic routines, which operate on data stored in preassigned memory locations. Data manipulation routines are also included, so that the microcomputer system can be operated much like a pocket calculator. It is important to remember that the monitor system is interrupt driven; thus after a keyboard initiated function has been completed, program control returns to the interrupted routine.

A. The Use of the Monitor System

In order to reduce the required number of keyboard elements, most keys have two assigned functions (see figure 43). The primary function, indicated on the lower portion of each key, is initiated by pushing the corresponding key. To initiate a secondary function, indicated on the upper part of the key, the function (FTN) key is first pushed, followed by the desired key. To minimize confusion, two important functions are selectable independent of the function (FTN) key. These functions, system reset (RESET) and display clear (CLEAR), are indicated by the legend's position in the center of the key.

System Reset

This routine is executed automatically when the power switch is turned on, and thus is usually only used when the monitor loses program control due to a software error. The function of the system reset routine is to initialize the monitor routine, clear the display, and enable the interrupt system. At this point, an 'r' is displayed and program control is transferred to an idle loop to await further instructions.

LDY 7	8	CGS 9	XXY E	STR F	RCL
+ 4	- 5	x 6	÷ C	HEX D	RUN
AF1 1	AF2 2	AF3 3	A	B	LAD
FTN	AF0 0	IO WR	IO RD	CLEAR	RESET

Figure 43 System Keyboard Layout

Data Entry Keys

The hexadecimal digits 0 to 9 and A to F are entered by depressing the proper keyboard element. As the corresponding digit is displayed on the LED display, it is also entered into memory. If too many characters are entered, the previously entered digits are overwritten, starting at the leftmost (most significant) valid location. In general, there are eight valid character locations; except when in the memory modification mode, where only the two hexadecimal digits (one byte) corresponding to the data word are alterable.

Load Memory Address

In order to load the memory address to the address register, the four hexadecimal digits defining the address are entered, followed by the depression of the load address (LAD) key. For example:

Key	Display
0	0
2	02
0	020
0	0200
LAD	0200.05

Note that the most significant digit is entered first. After the LAD key is depressed, the entered address is stored in the address register, and the content of the corresponding memory location is displayed (in this case 05).

Memory Read

Once the address register is loaded, the memory read (RD) key will increment the address register and the displayed address, The data stored at the new location are then displayed next to the new address.

Memory Write

In order to change the data stored at a particular location, the required address is first entered. The new data, as it is entered, will overwrite the old data on the display, but will not yet affect the old data in memory. Depressing the memory write (WR) key will then store the new data in memory, increment the displayed address, and display the contents of the new location. This data can then also be changed as above, or the next location can be read using the memory read function. For example:

Key	Display
LAD	2000.C3
3	2000.33
4	2000.34
WR	2001.00
RD	2002.47

If an error is made in entering new data and the error is noticed before the memory write key is pushed, the correct entry can simply be reentered, overwriting the incorrect one. Also, the memory read and memory write functions do not affect the initially entered address. Thus, depressing the load address key will reenter the original address, allowing one to easily review just entered data. For example:

Key	Display
LAD	2000.C3
3	2000.33
4	2000.34
2	2000.24
3	2000.23
WR	2001.00
LAD	2000.23

The Run Function

Depressing the RUN key after entering an address will cause a subroutine call of the program starting at the entered address. Upon encountering a return statement, program control will return to the execution of the routine which was interrupted by the key depression.

An error in the called routine can cause program control to be lost by the monitor, necessitating the use of the system reset (RESET) key. For instance, the execution of an interrupt disable, followed by a halt, will disable the monitor, as well as any other maskable interrupt initiated software.

I/O Port Read

The contents of any I/O port (if permitted by hardware) can be read by entering the two digit hexadecimal port address, followed by the function (FTN) and I/O read (IO RD) keys. The port address and the contents of that port are then displayed as follows:

Key	Display
0	0
5	05
FTN	05
IO RD	P05.FE

I/O Port Write

In order to output data to an I/O port, the port address is entered, followed by the two hexadecimal data digits. The function (FTN) and I/O port write (IO WR) keys are then pushed, as follows:

Key	Display
0	0
4	04
D	04D
C	04DC
FTN	04DC
IO WR	P04.DC

Eight LED's on the front panel are connected to port F6, such that a 0 will turn the corresponding light ON and a 1 will turn it OFF. Thus writing OF (hexadecimal for 0000 1111) to port address F6 will turn the four most significant (left) display locations on and the four least significant ones off.

Auxiliary Functions

Four auxiliary functions (AF0, AF1, AF2 and AF3) keys are provided. The purpose of these is to allow the user to call an often used subroutine without having to enter its starting address each time it is used. The function of these keys can be defined by software, or a 'forwarding address' to an elsewhere located routine can be entered from the keyboard. The 'forwarding addresses' are stored in the following memory locations:

Function	Address	Contents
AF0	2001	least significant byte
	2002	most significant byte
AF1	2004	least significant byte
	2005	most significant byte
AF2	2007	least significant byte
	2008	most significant byte
AF3	200A	least significant byte
	200B	most significant byte

As with the run function, the 'forwarding addresses' are accessed as subroutines.

B. Data Manipulation Routines

Calculator functions are provided to facilitate the convenient calculation of variables, and their entry into memory locations accessible to application programs. Numbers are entered and displayed in floating point decimal format, using a signed four digit mantissa and a signed one digit exponent. Although internal arithmetic operations allow decimal exponents ranging from +32 to -32, only exponents from +9 to -9 are entered or displayed. This limitation is imposed by the number of digits on the system display.

Data Entry

Decimal data is entered by entering the digits into the display via the keyboard. The function (FTN) key is then pushed, followed by the load (LDY) key. This enters the floating point decimal number, converts it to floating point binary, and stores it in the Y (result) register.

The entered number is interpreted as follows. The first four entered digits will be interpreted as a four digit mantissa, having a value between 1.000 and 9.999. In other words, a decimal point is implied after the first digit. The fifth entered digit, if there is one, will be interpreted to be the exponent. If a fifth digit is not entered, a zero exponent will be assumed. If either mantissa or exponent are to have a negative value, they must be preceded by a minus sign. The minus sign is entered by depressing the function key, followed by the change sign (CGS) key. Note that the minus sign is not counted as a digit.

Often the precision of the entered number does not require a four digit mantissa. If this is the case, the end of the mantissa string can be indicated by entering a minus sign if the exponent is to be negative, or an E if the exponent is to be positive. If the exponent is zero, entry can simply be terminated after the required number of mantissa digits. After the enter key is depressed, the floating point number will be displayed as it has been interpreted, as follows:

Key	Display
1	1
2	12
3	123
4	1234
5	12345
FTN	12345
LDY	1.234E 5
CLEAR	
FTN	
CGS	-
1	-1
2	-12
3	-123
FTN	-123
CGS	-123-
4	-123-4
FTN	-123-4
LDY	-1.230E-4

As illustrated in the example, an erroneous entry is removed by depressing the CLEAR key.

The following are several examples cf how data can be entered and how it will be interpreted:

Entered Data	Interpretation
1234	1.230E 4
1E4	1.000E 4
1-4	1.000E-4
12	1.200E 0
-1	-1.000E 0
12E3	1.200E 3

Arithmetic Operations

Arithmetic operations are performed using a reverse Polish notation. That is, a number is entered into the Y register using the Y register load function (FTN LDY). A second number is then entered into the display, and the desired operation is initiated. Depressing the FTN key, followed by the add, subtract, multiply or divide key, loads the presently displayed number into the X register (in floating point binary), and performs the desired operation on the number in the Y register. The result is then displayed and saved in the Y register, while the last used operant remains stored in the X register. By selecting the inverse operation without entering new data, the just performed operation can be undone.

The above mentioned operations are selected by depressing the function key, followed by the desired operation (+, -, x or ÷). The add function adds the X register to the Y register (leaving the result in the Y register); subtract, subtracts the X register from the Y register; multiply, multiplies the registers; and divide, divides the Y register by the X register. An actual calculation proceeds as follows:

Keys	Display
FTN,CGS,8,FTN,LDY	-8.000E 0
2,FTN,+	-6.000E 0
2E1,FTN,x	-1.200E 2
FTN,÷	-6.000E 0

Decimal to Binary Conversion

When a floating point decimal number is entered, it is converted to floating point binary before being stored. The floating point binary number stored in the Y register can be displayed in hexadecimal notation using the hexadecimal display function (FTN followed by HEX). The binary number consists of a two's complement 16 bit mantissa followed by a two's complement 8 bit exponent.

Exchange X and Y Registers

The exchange registers function (FTN followed by XXY) will exchange the contents of the X and Y registers. The new contents of the X register are then displayed in floating point decimal notation.

Storing and Recalling Data

Calculation results and entered data can be stored in one of 10 (0 to 9) memory registers. The number to be stored is taken from the Y register (the result register), while the recalled value is deposited in the X register (the operant register). After executing a store instruction the display is cleared, whereas after a recall instruction, the recalled number is displayed. Recalling data from a memory register does not alter the content of that register, only the storing of new data will change the memory.

Data are stored and recalled by first entering the one digit decimal number corresponding to the register, followed by the function (FTN) and store (STR) or recall (RCL) keys as illustrated below:

Key	Display
	1.230E 4 (Y)
9	9
FTN, STR	
9	9
FTN, RCL	1.230E 4 (X)

C. Operation of the DC Motor Control System

The microcomputer system, together with its DC motor interface module, constitutes a programmable, digital motor controller. An SCR circuit, located in the interface module, provides a full wave rectified 120 volt AC, variable duty cycle source, suitable for energizing the armature of a DC motor, or the field of a synchronous generator. A non variable, full wave rectified 120 volt source is also provided for the DC motor field and the synchronous generator field. A current to voltage transducer and an A/D converter, located in the interface module, allow the microprocessor to monitor the armature current. Angular velocity is determined by measuring the time interval between light pulses, modulated by a slotted disc which is mounted on the shaft of the motor generator set. Software routines are provided to allow the user to initialize the application module, start and stop the motor, alter control variables and set points, or to monitor the frequency of the 120 volt power grid. Error messages are provided to help the user avoid improper operating conditions.

Memory Register	Default Value	Variable
2	-	scaled time averaged current
3	-	present control output
4	0.02	proportional gain constant
5	0.0	integral gain constant
6	1000	RPM set point
7	-	present rpm
8	2.8	steady state current limit
9	-	present current

Figure 44 Motor Control Variables and Set Points

Motor Control Routine Initialization

Executing the motor control initialization routine will do the following:

1. The counter controlling the SCR circuit is disabled.
2. The following auxiliary functions are defined for later use:

   AF0 - stop motor

   AF1 - start motor

   AF2 - display frequency of power mains

3. The sample interval counter is defaulted to 13 (decimal), resulting in a sample interval of 100 msec. at 1000 RPM.
4. Default values for the control variables and set points

are loaded, as per figure 44 .

The initialization routine is called from the keyboard by loading its address (1000 hex) and pressing the load address and run keys. The display will be cleared, awaiting further instructions.

It must be kept in mind that when the microcomputer system is powered up, the SCR controlling circuit will initialize in an unpredictable state. Thus, to prevent damage to the motor control interface, or to the motor itself, the SCR ENABLE switch must be in the OFF position, from when power is first applied, until the motor control initialization routine has been executed.

Starting the Motor

Before starting the motor, the light source for the digital tachometer should be activated. The SCR ENABLE switch should then be turned ON and the motor started by pressing the function key (FTN), followed by the auxiliary function 1 (AF1) key.

Stopping the Motor

The motor can be stopped under program control using auxiliary function 0 (AF0). If for some reason program control is lost, the motor can also be brought to a stop by turning the SCR ENABLE switch OFF.

Altering and Reading Control Variables and Set Points

The following motor control variables and set points are readily altered by the system user:

1. The proportional gain constant.
2. The integral gain constant.
3. The RPM setpoint.
4. The continuous armature current limit.
5. The sample interval counter.

All of the above variables, excluding the last, are changed by loading the new variable into the corresponding memory register , as per figure 44. The sample interval counter is altered by loading the new value, in hexadecimal, into memory location 2065 hex using the system monitor.

The following control outputs are also available to the user:

1. The present RPM.
2. The present armature current.
3. The present, scaled, time averaged armature current.
4. The present control output.

These are accessed by recalling the appropriate memory register, as indicated in figure 44.

It should be kept in mind that since the microcomputer system is interrupt driven, it is entirely possible to alter control variables or set points, or to use the calculator or monitor functions, while the motor control program is active.

Continuous Monitoring of Variables

It is often desirable to continuously display one of the system parameters. The program listed in figure 45 reads the desired variable from the memory register, converts it to a floating point decimal number, and writes it to the system display; updating the variable every two seconds.

In order to take advantage of this program, it must first be entered into memory using the keyboard monitor, starting at location 2400 hex. The program is then initiated by loading the starting address and pressing the RUN key. The motor can be started either before or after initializing the display program, by using the proper auxiliary function.

Execution of the variable display program is terminated by depressing the clear key. New commands can then be entered, or the program itself can be modified. For example, altering memory location 2419 hex will alter the update interval, while changing location 2411 hex will specify the display of a different memory register. During use, this program replaces the idle loop as the background

```
                      ******************
                      * MEMORY DISPLAY *
                      *   ROUTINE      *
                      ******************

            CRCLF EQU 0C1BH
            DDAT7 EQU 2018H
            DISPF EQU 2066H    DISPLAY ENABLE FLAG

2400 F5     START:PUSH PSW     SAVE INT REGISTERS
2401 C5           PUSH B
2402 E5           PUSH H
2403 FB           EI           ENABLE INTERRUPTS
2404 3EFF         MVI A,0FFH
2406 326620       STA DISPF    SET ENABLE FLAG
2409 3A6620 LOOP0:LDA DISPF    LOAD DISPLAY FLAG
240C A7           ANA A        TEST
240D CA2F24       JZ  ENDIT    EXIT IF RESET
2410 3E07         MVI A,07H    LOAD MEMORY REGISTER
                               TO BE DISPLAYED
2412 321820       STA DDAT7    STORE IN MEMORY
                               ADDRESS POINTER
2415 CD1B0C       CALL CRCLF   CALL ROUTINE TO
                               DISPLAY DATA
2418 0604         MVI B,04H    LOAD COUNTER FOR NUMBER
                               OF HALF SECOND DELAY LOOPS
241A 2100F4 LOOP1:LXI H,0F400H LOAD HALF SECOND
                               DELAY COUNT
241D 2B     LOOP2:DCX H        DECREMENT
241E 7D           MOV A,L
241F A7           ANA A        TEST LS BYTE
2420 C21D24       JNZ LOOP2    REPEAT IF NOT ZERO
2423 7C           MOV A,H
2424 A7           ANA A        TEST MS BYTE
2425 C21D24       JNZ LOOP2    REPEAT IF NOT ZERO
2428 05           DCR B        DECREMENT HALF
                               SECOND COUNTER
2429 C21A24       JNZ LOOP1    REPEAT IF NOT DONE
242C C30924       JMP LOOP0    DELAY EXPIRED,
                               DISPLAY NEW DATA
242F E1     ENDIT:POP H        RESTORE INTERNAL REGS
2430 C1           POP B
2431 F1           POP PSW
2432 C9           RET          RETURN
```

Figure 45   Routine for Continuous Memory Register Display

routine, thus the interrupt driven motor control routine should not be interfered with.

Monitoring the Power Mains Frequency

Synchronizing pulses provided by the motor interface module can also be used to determine the frequency of the power grid. Once the motor initialization routine has been executed, the power mains frequency can be displayed by depressing the function key, followed by auxiliary function 2 (AF2). It is not necessary to turn the SCR ENABLE switch ON in order to execute this routine.

D. System Error Messages

In order to give the user a fighting chance in debugging system software problems, a number of error exits are strategically located throughout the system software. When such an error occurs, the microcomputer will display an error code (for example: Er 13), execute a halt command, and await further instructions from the keyboard.

The error codes are defined as follows:

1. An interrupt has occurred for which no service routine is defined. An undefined interrupt vector at the beginning of memory, or noise on the interrupt lines are likely causes.
2. A key has been depressed for which no service routine is defined. There is also a possibility that an erroneous key code was returned from the 8279 on the

keyboard/display interface board. (Also see error 3.)

3. A keycode corresponding to a non existing key was returned from the 8279 on the keyboard/display board.
4. A binary exponent overflow has occurred in the floating point multiplication routine.
5. A binary exponent underflow has occurred in the floating point multiplication routine.
6. A binary exponent underflow has occurred in the floating point normalization routine.
7. A binary exponent overflow has occurred in the floating point division routine.
8. An attempt was made to divide by zero in the floating point division routine.
9. Binary exponent overflow in the floating point division routine.
10. Binary exponent underflow in the floating point division routine.
11. An out of range memory register address was encountered in the execution of the data store instruction.
12. An out of range memory register address was encountered in the execution of the data recall function.
13. An auxiliary function key was used without first entering the 'forwarding address'.
14. A decimal exponent of magnitude greater than 99 was encountered in the floating point binary to decimal conversion routine.
15. A two digit decimal exponent was encountered in the

decimal display routine.

16. A decimal exponent overflow has occurred after exponent roundoff in the decimal display routine.
17. Undefined.
18. Undefined.
19. Undefined.
20. Data contained in a memory location does not agree with data that has just been entered using the memory write function. The most likely cause of this problem is an attempt to write to a read only memory location.
21. An attempt has been made to start the DC motor without specifying the maximum operating current.
22. An RPM set point has not been entered.
23. Neither a proportional nor and integral gain constant has been defined.
24. The instantaneous current limit has been exceeded.
25. The time averaged current limit has been exceeded.

The above error definitions are only a brief summary; more detailed information can be found in the following software documentation.

V. System Hardware

In this chapter a detailed description of the system hardware is presented. The design of the circuitry was governed by the following considerations:

1. The system should be composed of several functional modules to facilitate troubleshooting and permit modification and expansion.
2. An attempt should be made to obtain reasonable power dissipation in the digital circuitry, through the use of low power Schottky logic where possible.
3. A reasonable amount of self protection should be included in the design. This includes error checks in software, the use of proper fusing, and optical isolation between the microcomputer system and the motor interface module.

A. Bus Structure

The bus system serves two functions: the distribution of regulated power; and the transmission of data, address, and control signals between the individual circuit cards comprising the system. Several edge connector elements are used in parallel for the power supply connections to reduce the effects of contact resistance. The definition of the individual logic signals making up the system bus is a compromise between too complex a data encoding scheme and too many individual control lines. The bus lines are allocated as follows:

1. Regulated +5 volt supply; four parallel connectors are used.	51. Regulated +5 volts
2. Regulated +5 volts	52. Regulated +5 volts
3. Left blank for isolation	53. Left blank for isolation
4. D0 - bidirectional 8 bit data bus; all elements inverted.	54. D1
5. D2	55. D3
6. D4	56. D5
7. D6	57. D7
8. A0 - 16 bit address bus; all elements inverted.	58. A1
9. A2	59. A3
10. A4	60. A5
11. A6	61. A7
12. A8	62. A9

13. A10	63. A11
14. A12	64. A13
15. A14	65. A15
16. SOD - Serial data output line; inverted.	66. SID serial data input line; inverted.
17. AV - Least significant address byte valid; active low.	67. EINT - encoded interrupt (EI0 to EI7) from 8214; active low.
18. RD - Read ANDed with clock phase 2; active high.	68. INTA - Read signal for interrupt instruction; active high.
19. WR - Write ANDed with clock phase 2; active high.	69. IOM - I/O address (low), or memory address (high).
20. S0 - Inverted; S0 and S1 provide encoded advance read/write signals.	70. S1 - Inverted.
21. Reserved for external clock input.	71. COUT - Phase 2 clock output; inverted.
22. RIN - Reset input; active low.	72. ROUT - Reset output, active low.
23. HIN - CPU hold request; active low.	73. HOUT - Hold acknowledge output; active low.
24. RDY - Memory not ready; active low.	74. OF - Opcode fetch - used to trigger the logic analyzer; active low.
25. Spare	75. Spare
26. Spare	76. Spare
27. Spare	77. Spare
28. Spare	78. Spare
29. Spare	79. Spare
30. Spare	80. Spare
31. Spare	81. Spare
32. Spare	82. Spare

33. Spare
34. Spare
35. Spare
36. Spare
37. Spare
38. II1 - On chip interrupt; RST 5.5; active low.
39. II3 - RST 7.5; active low.
40. EI0 - External interrupt lines to 8214; INT0; active low.
41. EI2 - INT2
42. EI4 - INT4
43. EI6 - INT6
44. HFC - High frequency clock out; 1 MHz.
45. Left blank for isolation.
46. Regulated -5 volt supply; two parallel connectors are used.
47. Regulated +12 volt supply; two parallel connectors are used.
48. Spare
49. Ground; four parallel connectors are used.
50. Ground

34. Spare
84. Spare
85. Spare
86. Spare
87. Spare
88. II2 - RST 6.5; acitve low.
89. NMI - Non maskable interrupt; TRAP; active low.
90. EI1 - INT1.
91. EI3 - INT3.
92. EI5 - INT5.
93. EI7 - INT7
94. LFC - Low frequency clock out; 100 kHz.
95. Left blank for isolation.
96. Regulated -5 volts.
97. Regulated +12 volt supply.
98. Spare
99. Ground
100. Ground

More information on the individual bus signals can be found by referring to the microprocessor board description.

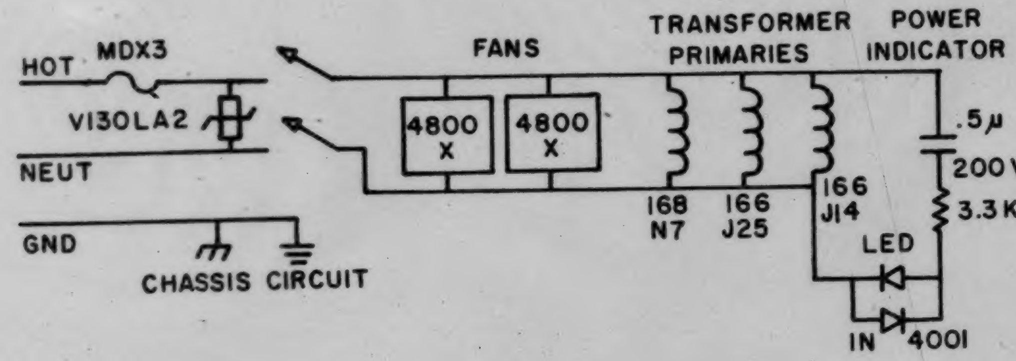

HOT
MDX3
V130LA2
NEUT
GND
CHASSIS
CIRCUIT
FANS
4800 X
4800 X
TRANSFORMER PRIMARIES
168 N7
166 J25
166 J14
POWER INDICATOR
.5µ
200 V
3.3 K
LED
IN 4001

Figure 46 AC Wiring Diagram

B. AC Wiring

The AC input is limited by a 3 amp, slow blowing fuse. The fuse ampacity is large when only steady state current is considered; but it is required to supply the inrush current drawn by the large power supply capacitors. Protection from incoming transients is provided by a transient suppressor connected between the hot and neutral conductors. A double pole switch is used to provide isolation on both the hot and neutral sides of the primary circuit.

The power on indicator circuit [41] consists of an LED in series with a current limiting impedance connected accross the 120 V source. The diode conducts when the LED is off, preventing reverse breakdown of the LED. A capacitor forms part of the series impedance, reducing the real power dissipated as heat.

C. System Power Supplies

Regulated DC power is supplied at three voltages: +12 volts at 1 amp; -5 volts at 1 amp; and +5 volts at 3 amps. The +12 and -5 volt supplies are only required for the 2708 memory ICs, and are adequate even for an expanded system. The three amp +5 volt supply however, will nct support significant expansion beyond the existing system, thus room has been left for the insertion of a more substantial supply. The existing transformer and filter capacitors are adequate for a +5 volt supply of up to 14 amps. The power supply circuitry is illustrated in figures 47 to 49.

System Cooling

The TO-220 packages housing the -5 and +12 volt regulators are cooled using Wakefield 680-.75 heat sinks. The +5 volt regulator TO-3 pack requires a 680-1.25 heat sink due to the higher current passing through it. All three regulators will supply full rated current without exceeding maximum junction temperatures with only natural convecitve cooling. However, when the CPU enclosure is fully assembled, natural convection is not possible, thus two Pamotor 4800X fans are used to force air through the enclosure, cooling both the power supplies and the digital circuitry.

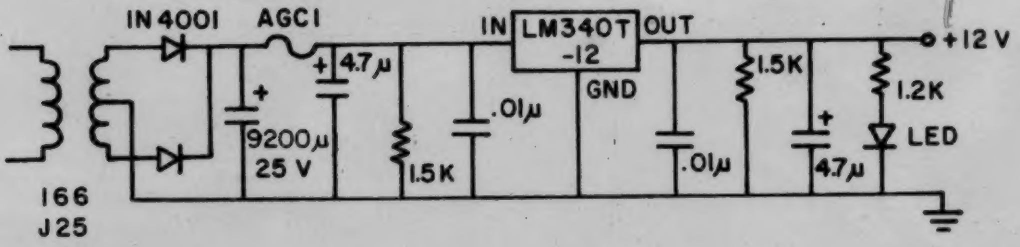

IN4001
AGC1
4.7μ
9200μ
25 V
1.5K
.01μ
IN
LM340T
-12
OUT
GND
.01μ
1.5K
4.7μ
1.2K
LED
+12 V
166
J25

Figure 47 Positive 12 Volt Regulator Circuit

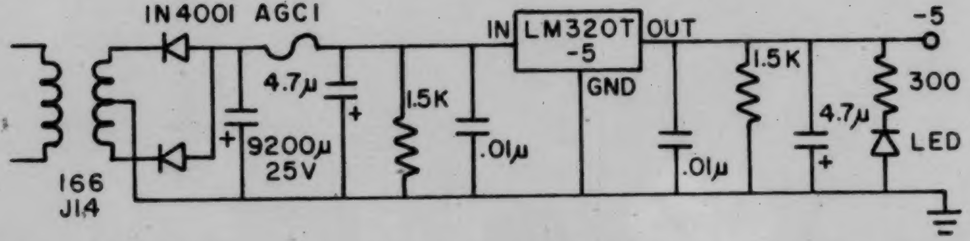

IN4001
AGC1
4.7μ
9200μ
25V
1.5K
.01μ
IN
LM320T
-5
OUT
GND
.01μ
1.5K
4.7μ
300
LED
-5
166
J14

Figure 48 Negative 5 Volt Regulator Circuit

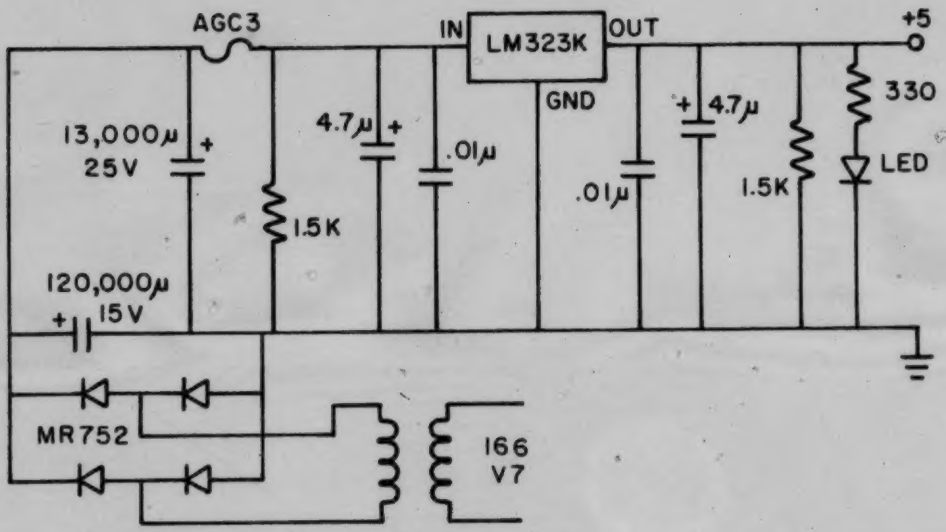

AGC3
13,000μ
25V
1.5K
4.7μ
.01μ
IN
LM323K
OUT
GND
.01μ
4.7μ
1.5K
330
LED
+5
120,000μ
15V
MR752
166
V7

Figure 49 Positive 5 Volt Regulator Circuit

D. Microprocessor Board

The main microprocessor board consists of two functional blocks:

- the microprocessor chip and the associated circuitry interfacing it to the bus; and
- the circuitry required to interface the Intel 8279 keyboard/display chip to the system bus.

The latter interface will also accomodate the future addition of an Intel 8253 timer IC, and an Intel 8214 interrupt decoder circuit.

Figures 50 to 53 illustrate the circuitry located on the microprocessor board, while figure 52 provides the corresponding component list. The alphabetic IC position code given on the component list refers to the position of the IC package on the wirewrap board, relative to the alphabetic grid printed along the edges of the board.

CPU Control Signals

Figure 50 illustrates the interface circuitry between the Intel 8085 microprocessor (IC BZ) and the system bus. The interface circuitry is designed such that a CPU hold acknowlege will place all bus drivers affecting the system address, data, and control line into a high impedance state. This will allow the future addition of direct memory access. The use of 8T series high impedance bus drivers and receivers allows the full expansion of the system to fill the 11 provided card slots, as well as permitting the

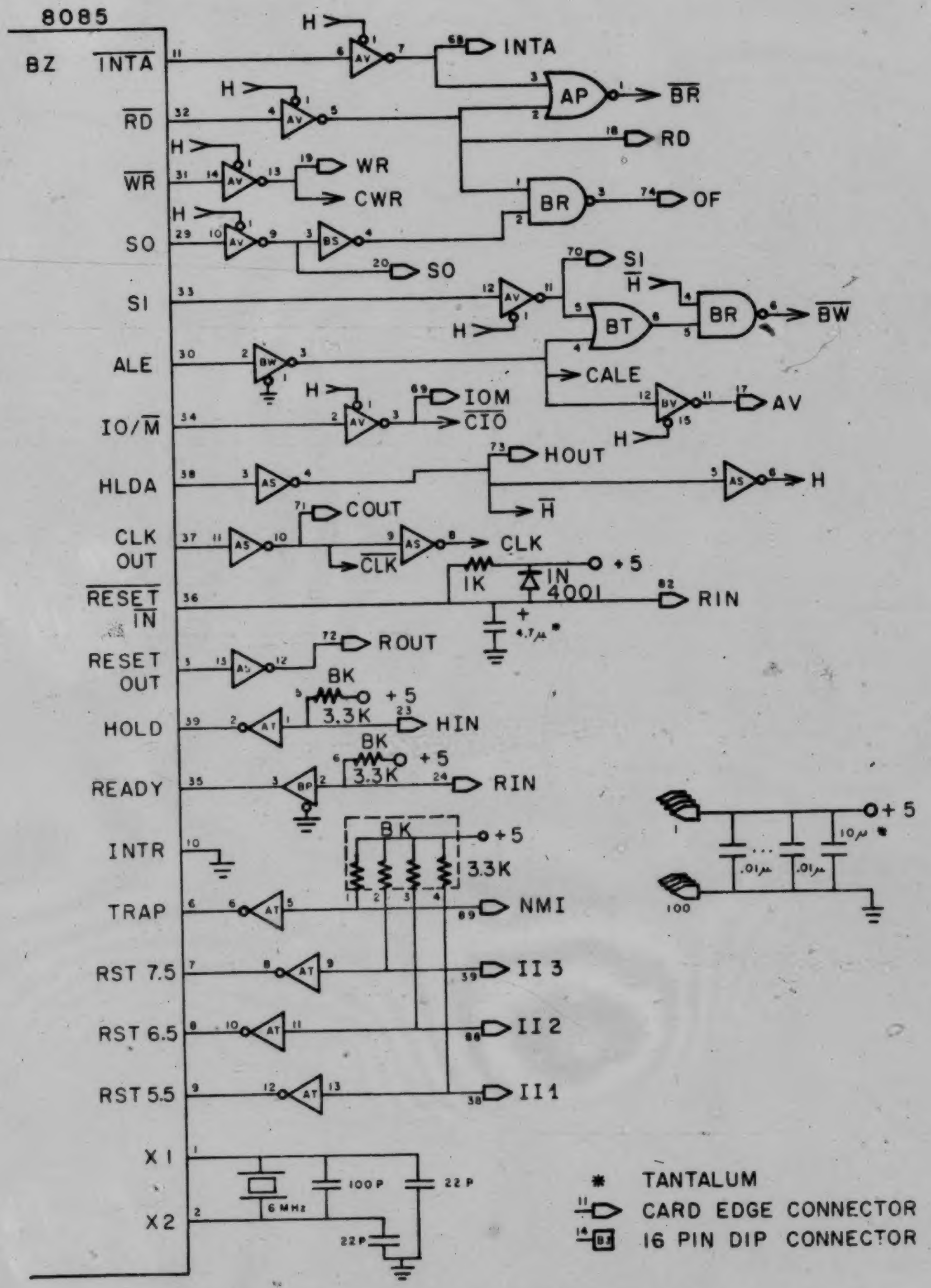

8085
BZ
INTA
RD
WR
SO
SI
ALE
IO/M
HLDA
CLK OUT
RESET IN
RESET OUT
HOLD
READY
INTR
TRAP
RST 7.5
RST 6.5
RST 5.5
X1
X2
H
AV
AP
BR
BS
BT
BW
BV
AS
AT
BP
BK
INTA
BR
RD
WR
CWR
OF
SO
SI
BW
CALE
AV
IOM
CIO
HOUT
H
COUT
CLK
CLK
1K
IN 4001
+5
RIN
4.7µ *
ROUT
3.3K
HIN
RIN
3.3K
NMI
II 3
II 2
II 1
6 MHz
100 P
22 P
22 P
10µ *
.01µ
.01µ
1
100
* TANTALUM
CARD EDGE CONNECTOR
16 PIN DIP CONNECTOR

Figure 50 Main CPU Board - Control Signals

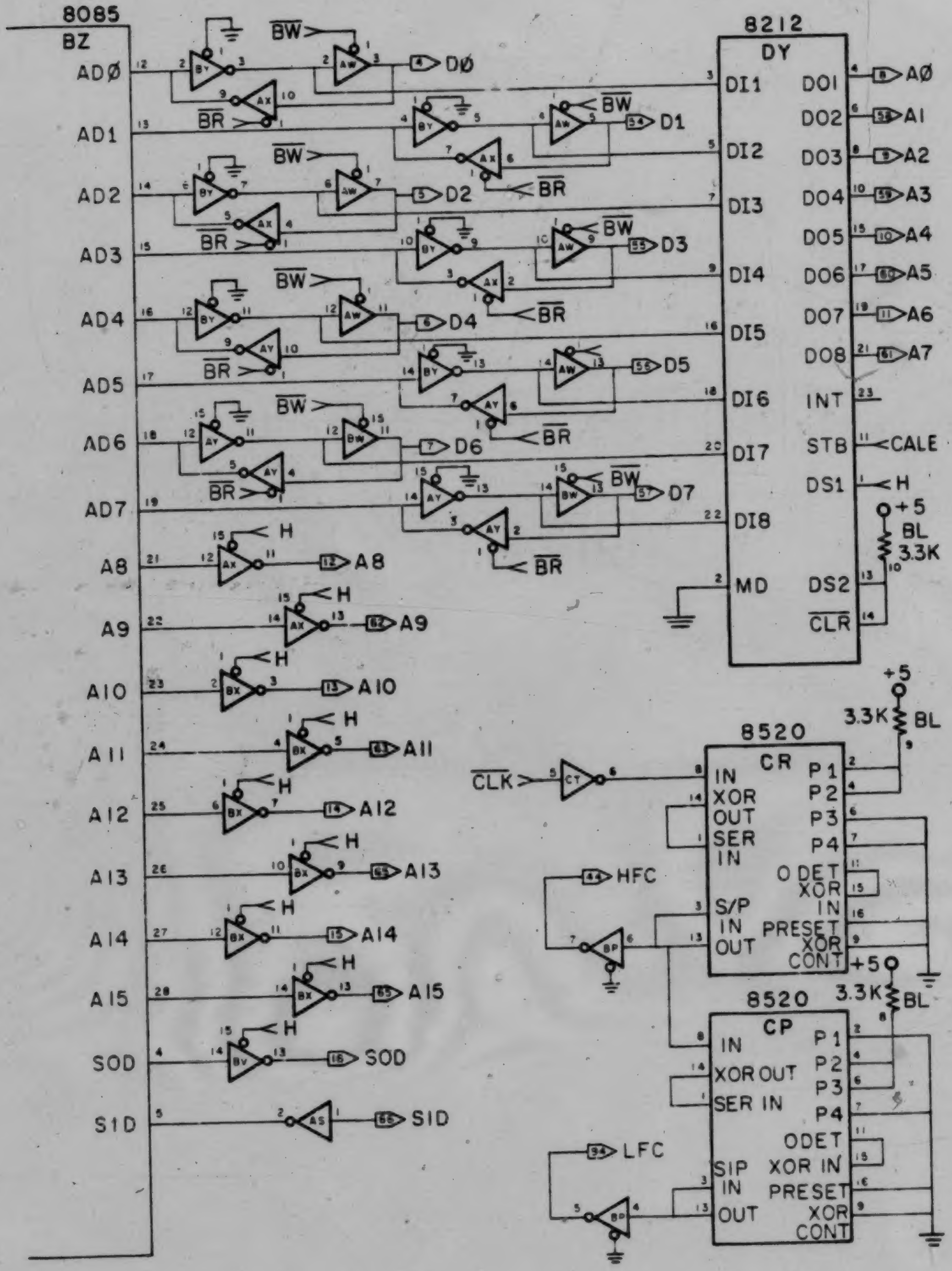

8085
BZ
AD0
AD1
AD2
AD3
AD4
AD5
AD6
AD7
A8
A9
A10
A11
A12
A13
A14
A15
SOD
SID
BW
BR
D0
D1
D2
D3
D4
D5
D6
D7
H
8212
DY
DI1
DI2
DI3
DI4
DI5
DI6
DI7
DI8
MD
DO1
DO2
DO3
DO4
DO5
DO6
DO7
DO8
INT
STB
DS1
DS2
CLR
A0
A1
A2
A3
A4
A5
A6
A7
CALE
+5
BL
3.3K
8520
CR
CP
CLK
IN
XOR
OUT
SER
IN
S/P
IN
OUT
P1
P2
P3
P4
O DET
XOR
IN
PRESET
XOR
CONT
HFC
LFC

Figure 51   Main CPU Board - Address and Data lines

IC Position	IC Type	Power	Gnd	Misc Connections
AJ	switches	-	-	gnd: 9 to 16
AP	74LS02	14	7	
AR	74LS08	14	7	
AS	74LS04	14	7	
AT	74LS04	14	7	
AV	8T96	16	8	gnd: 15
AW	8T95	16	8	gnd: 15
AX	8T98	16	8	
AY	8T98	16	8	
BK	3.3K Rs	14	-	
BL	3.3K Rs	14	-	
BM	3.3K Rs	14	-	
BP	8T95	16	8	gnd: 1, 15
BR	74LS00	14	7	
BS	74LS04	14	7	
BT	74LS32	14	7	
BV	8T98	16	8	
BW	8T97	16	8	gnd: 1
BX	8T96	16	8	gnd: 15
BY	8T96	16	8	gnd: 15
BZ	8085	40	20	
CK	8160	16	8	
CL	74LS20	14	7	
CP	8520	5	12	
CR	8520	5	12	
CT	74LS04	14	7	
CV	74LS04	14	7	
CW	74LS04	14	7	
CX	8T96	16	8	gnd: 15
CY	8T98	16	8	
DY	8212	24	12	

Figure 52 CPU Board IC's

addition of bus pullup resistors.

The 8085 provides separate read and write signals which are coupled to the bus through inverting buffers. IC AP ORs the read and interrupt acknowlege signals to produce the bus read signal; $\overline{BR}$. This is used to enable the bus receivers for either a normal read, or to read the interrupt instruction which is required if the INTR pin on the 8085 is used. Since the duration of the read and write pulses are too short for slower memory IC's, such as the 2708's, S0 and S1 are also coupled to the bus. These two signals can be decoded to provide an advanced read or write signal for use with slower devices.

The bus write signal ($\overline{BW}$) enables the data bus drivers in the write mode. ALE and S1 are used to derive this signal, such that the data bus is active not only for the entire data valid period, but also while ALE is valid, so that the lower order address byte can also be obtained from the data bus. This has been done to maintain system compatability with the newer 8085 support chips, many of which require this multiplexed address and data.

The reset input, the hold and memory ready inputs, and the system interrupts (TRAP and RST 5.5 to 7.5) have been buffered such that the input to the board is pulled high (inactive) with a resistor; providing a wired OR configuration. An active state can then be obtained by pulling the proper bus line low.

The system clock consists of a 6 MHz crystal connected to the 8085's on chip oscillator circuit. A 100 picofarad capacitor is used to trim the frequency, such that a deviation of less than 0.01% is obtained. The CPU chip's 3 MHz clock output is divided down, using two 8520 counter ICs, to provide 1 MHz and 100 KHz clock outputs.

Address and Data Lines

Address and data buffering between the microprocessor and the system bus is illustrated in figure 51. The 8085 outputs the least significant address byte on the data pins during the first phase of a machine cycle. An 8212 (IC DY) is used as a transparent latch to hold this address byte for the duration of the machine cycle. The most significant address byte is stable for the duration of the machine cycle, and thus only needs to be buffered. The data line buffers, as mentioned previously, are enabled in the bus write mode for the entire write cycle. For a read cycle, they are enabled in the bus read mode only for the duration of clock phase ø2.

E. Keyboard and Display Interface

Also located on the CPU board is the circuitry required to interface several peripheral chips to the system bus (see figure 53). Presently only an 8279 display/keyboard interface IC is used; however, provisions have been made to add other devices, such as an 8214 interrupt decoder and an 8253 timer/counter IC.

A bidirectional data bus is provided (PD0 to PD7) for use with these peripheral devices. The bus buffers are active in the CPU read (bus write) mode for clock phase two of a CPU I/O read cycle if the board select signal (PBA) is active; or for an interrupt acknowledge. At all other times, the bus drivers are in a high impedance state.

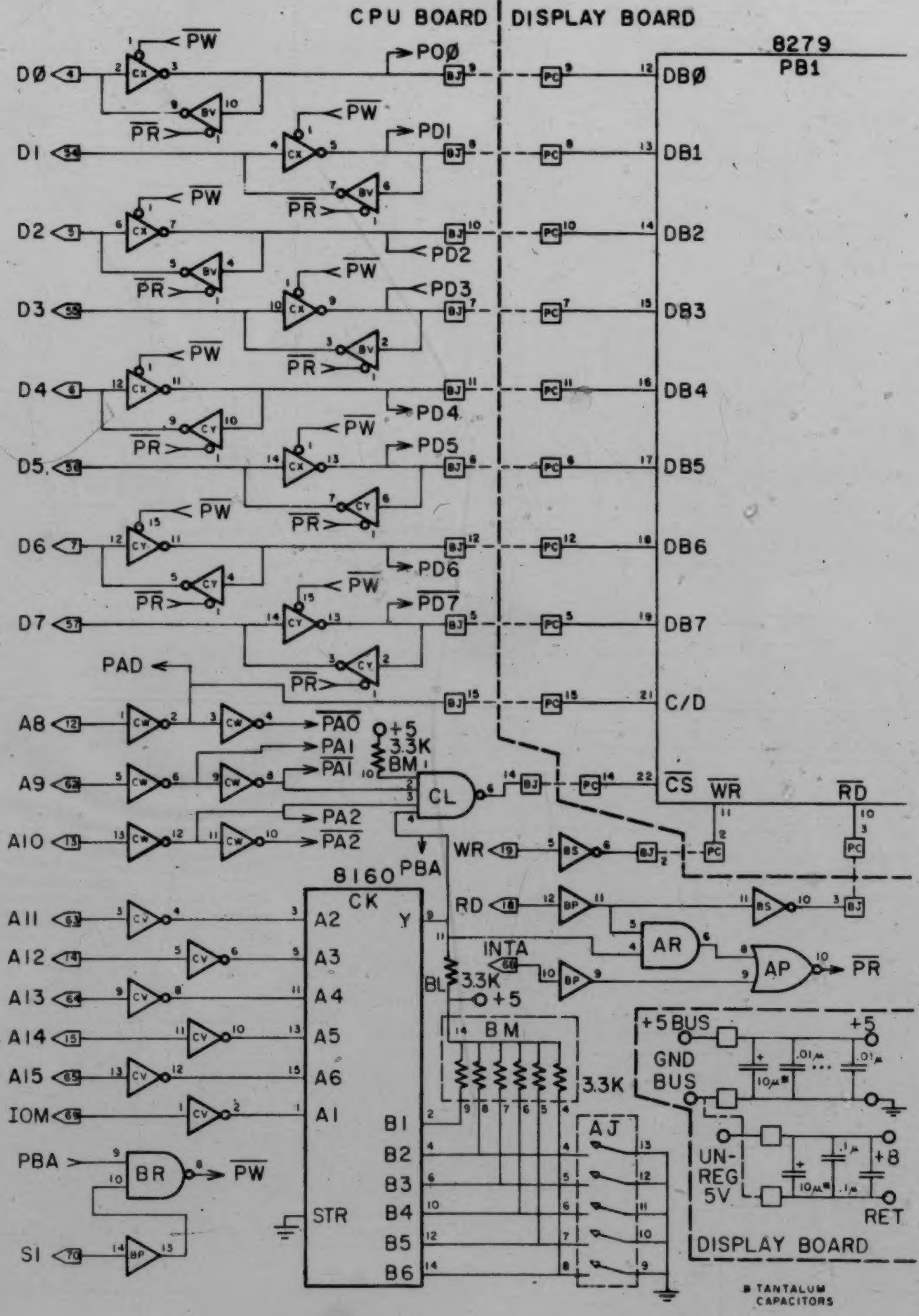

CPU BOARD
DISPLAY BOARD
8279
PB1
D0
D1
D2
D3
D4
D5
D6
D7
PW
PR
PD0
PD1
PD2
PD3
PD4
PD5
PD6
PD7
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
C/D
CS
WR
RD
PAD
A8
A9
A10
PA0
PA1
PA2
+5
3.3K
BM
CL
WR
RD
INTA
AR
AP
PR
PBA
8160
CK
A11
A12
A13
A14
A15
IOM
A2
A3
A4
A5
A6
A1
Y
B1
B2
B3
B4
B5
B6
STR
BL
3.3K
+5
BM
3.3K
AJ
PBA
BR
PW
S1
+5 BUS
GND BUS
+5
10µ
.01µ
UN-REG 5V
+8
RET
.1µ
DISPLAY BOARD
TANTALUM CAPACITORS

Figure 53 Peripheral IC Interface to System Bus

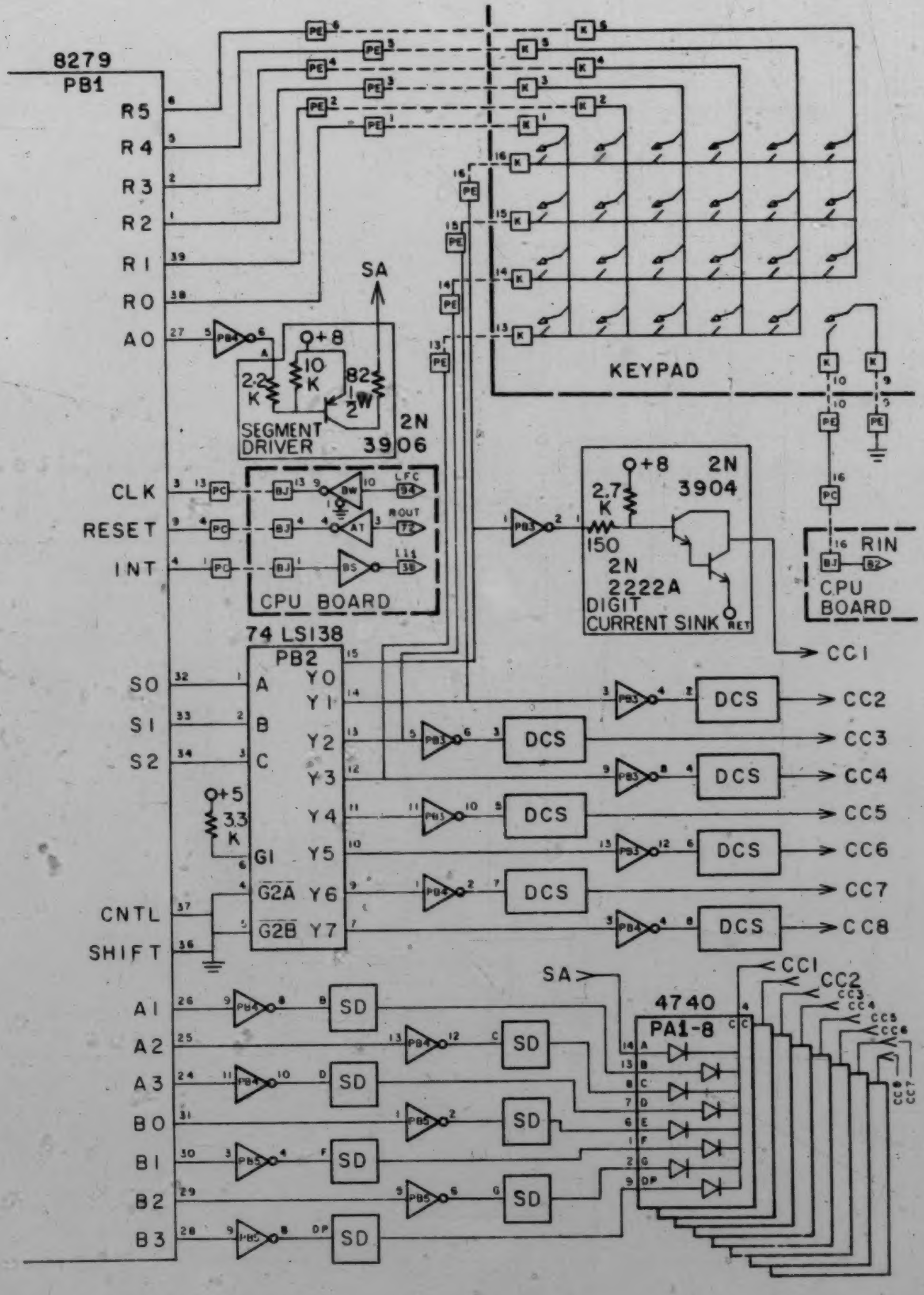

8279
PB1
R5
R4
R3
R2
R1
R0
A0
CLK
RESET
INT
S0
S1
S2
CNTL
SHIFT
A1
A2
A3
B0
B1
B2
B3
SA
+8
2.2 K
10 K
82 1/2 W
SEGMENT DRIVER
2N 3906
CPU BOARD
74 LS138
PB2
A
B
C
G1
G2A
G2B
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
+5
3.3 K
KEYPAD
2N 3904
2.7 K
150
2N 2222A
DIGIT CURRENT SINK
RIN
CPU BOARD
DCS
CC1
CC2
CC3
CC4
CC5
CC6
CC7
CC8
SD
4740
PA1-8

Figure 54 System Display and Keypad

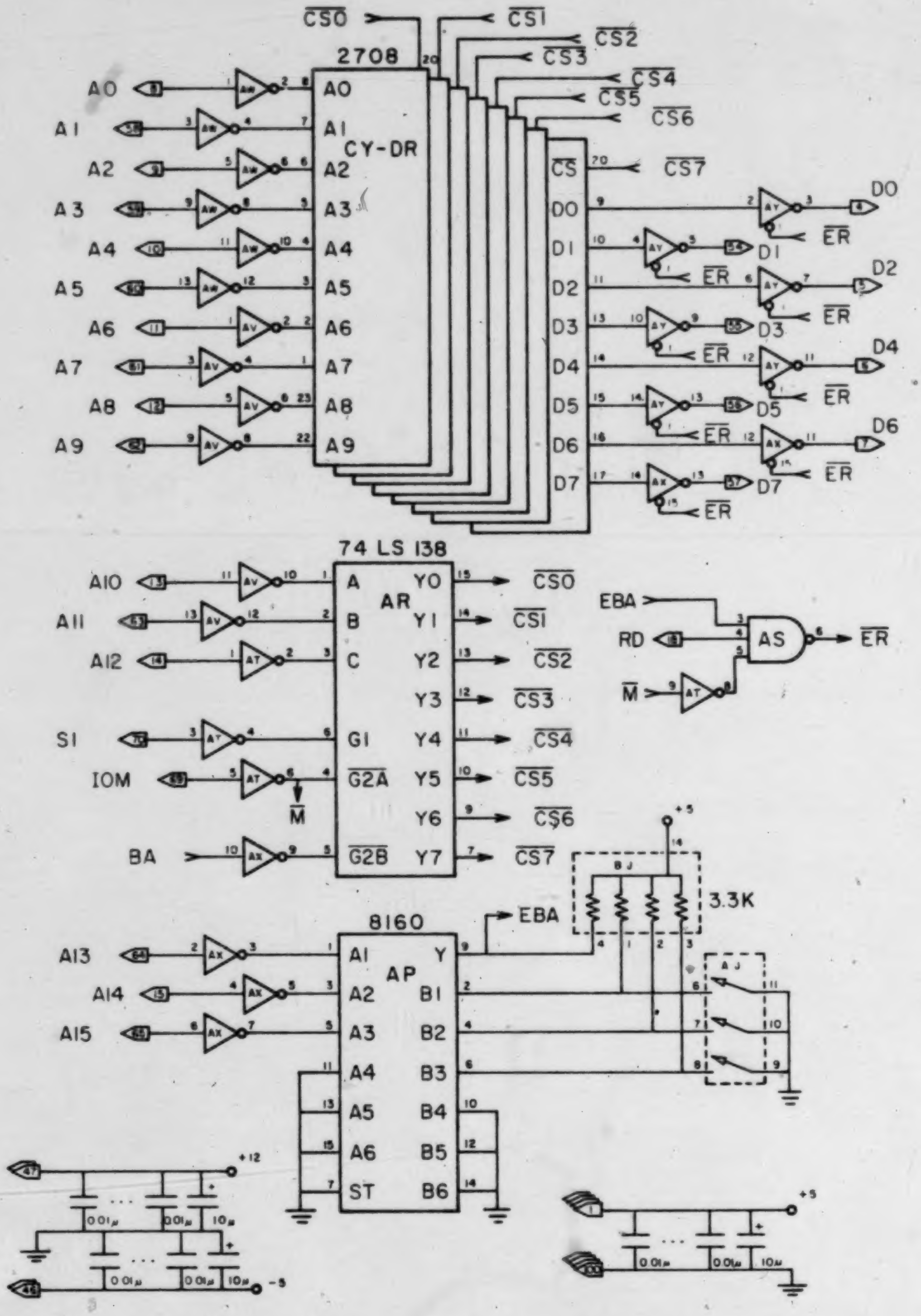
IC Position	IC Type	Power	Gnd	Misc Connections
PB1	8279	40	20	
PB2	74LS138	16	8	gnd: 4, 5
PB3	7406	14	7	
PB4	7406	14	7	
PB5	7406	14	7	

Figure 55 Display Board IC's

During a CPU I/O write cycle, the bus receivers are active in the CPU write (bus read) mode for the entire cycle.

The five most significant bits of the 8 bit I/O address are compared to five switch inputs by IC CK to derive a board select signal, PBA. This, and the three least significant address bits, provided both inverted and noninverted, can then be used to derive the required chip select signals for the individual support chips. The 8279 thus occupies I/O addresses nnnn n100 and nnnn n101 (binary), where n is selected by the switches (IC AJ).

The required signals are then brought to the 8279, located on the system display board, via a 16 conductor ribbon cable (figure 54). A 3 to 8 decoder is used to decode the S0 to S2 outputs of the 8279 (not to be confused with the S0 and S1 outputs of the 8085) in order to select an individual row of the keyboard matrix, as well as an individual seven segment display digit. The keyboard matrix columns are scanned for a switch closure by the 8279's return lines; R0 to R5.



CS0
CS1
CS2
CS3
CS4
CS5
CS6
CS7
2708
CY-DR
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
CS
D0
D1
D2
D3
D4
D5
D6
D7
ER
74 LS 138
AR
A
B
C
G1
G2A
G2B
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
A10
A11
A12
S1
IOM
M
BA
EBA
RD
AS
+5
BJ
3.3K
AJ
8160
AP
Y
A13
A14
A15
B1
B2
B3
B4
B5
B6
ST
+12
-5
0.01μ
10μ

Figure 56 Read Only Memory Board

IC Position	IC Type	Power	Gnd	Misc Connections
AJ	switches	-	-	gnd: 8 to 11
AP	8160	16	8	
AR	74LS138	16	8	
AS	74LS10	14	7	
AT	74LS04	14	7	
AV	74LS04	14	7	
AW	74LS04	14	7	
AX	8T98	16	8	gnd: 1
AY	8T96	16	8	gnd: 15
BJ	3.3K Rs	14	-	
CY to DR	2708	24	12	gnd: 18; +12: 19; -5: 21

Figure 57 ROM Board ICs

Display data is output through the data pins A0 to A3 and B0 to B3; buffered through high voltage open collector inverters. The inverters in turn enable the proper current sourcing transistors, which enable the required display segments. The currently active output of the 3 to 8 decoder, through a similar inverter/transistor circuit, sinks the common cathode current from the proper display IC.

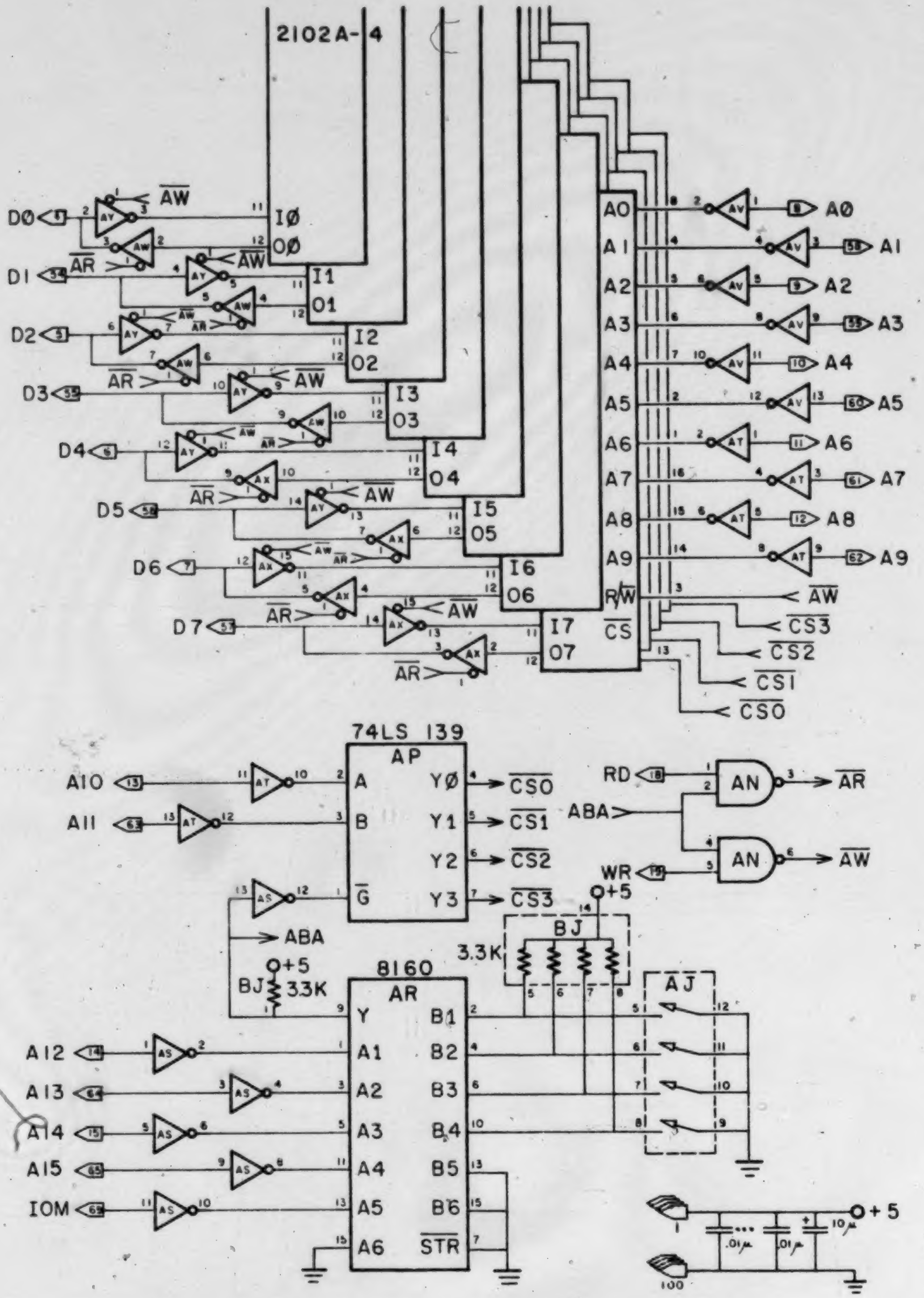

2102A-4
D0
D1
D2
D3
D4
D5
D6
D7
AW
AR
AY
AX
I0
O0
I1
O1
I2
O2
I3
O3
I4
O4
I5
O5
I6
O6
I7
O7
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
R/W
CS
AV
AT
CS3
CS2
CS1
CS0
74LS 139
AP
A10
A11
A
B
G
Y0
Y1
Y2
Y3
ABA
AS
RD
WR
AN
+5
BJ
3.3K
33K
8160
AR
Y
A12
A13
A14
A15
IOM
A1
A2
A3
A4
A5
A6
B1
B2
B3
B4
B5
B6
STR
AJ
.01µ
10µ
1
100

Figure 58   Random Access Memory Board

1

IC Position	IC Type	Power	Gnd	Misc Connections
AJ	switches	-	-	gnd: 8 to 11
AP	74LS139	16	8	
AR	8160	16	8	
AS	74LS04	14	7	
AT	74LS04	14	7	
AV	74LS04	14	7	
AW	8T98	16	8	
AX	8T98	16	8	
AY	8T96	16	8	gnd: 15
BJ	3.3K Rs	14	-	
BY to DK	2102A-4	10	9	

Figure 59 RAM Board ICs

F. Read Only Memory

The read only memory board (figures 56 and 57) is designed to hold up to eight 2708 EPROMs, providing 8K bytes of permanent memory (1 K of memory represents 1024 bytes). The three most significant bits of the board address are switch selectable. This selected board address is compared to the three most significant address bits by IC AP. If a match is obtained, the board address signal, EBA becomes active. The board address signal, together with the memory address signal ($\overline{M}$) and the advanced read signal (S1) enable the output of the 2708 selected by address lines A10 to A12 through IC AR. The data bus drivers are enabled during phase two of the system clock by the bus write signal ($\overline{ER}$) generated by IC AS.

G. Random Access Memory

The random access memory board (see figures 58 and 59) consists of sixteen 2102A-4 memories, providing 2K eight bit words of memory. Space on the board and address decoding has been provided for the addition of a further 2K bytes of RAM. The first 1024 bytes of memory, provided by ICs BY to BP (corresponding to D0 to D7) are selected by $\overline{CS0}$; the second 1K, consisting of ICs CY to CP, are selected by $\overline{CS1}$; the third 1K, DY to DP, by $\overline{CS2}$; and the final 1K, ICs BN to BK and CN to CK, are selected by $\overline{CS3}$.

The board address is selected by switch package AJ, and is compared to the four most significant address bits by IC AR. IC AP decodes address bits A10 and A11 to select the desired 1024 byte block of memory. If the board address, ABA, is valid, the bus buffers are enabled in the bus write (CPU read) mode by the memory read signal ($\overline{AR}$). During the active portion of the CPU write signal, the board write signal, $\overline{AW}$, enables the data bus buffers in the bus read (CPU write) mode, and the 2102A-4s in the write mode.

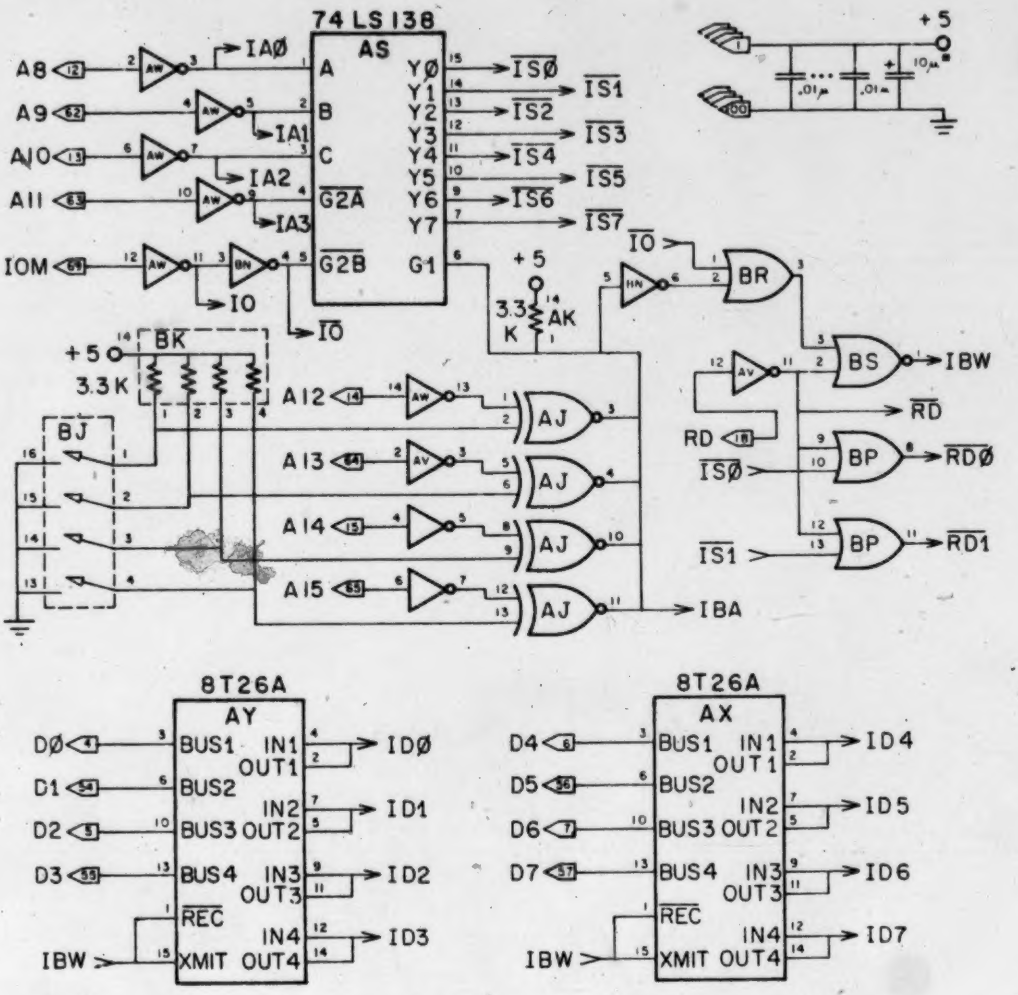

74 LS 138
AS
A8
A9
A10
A11
IOM
IA0
IA1
IA2
IA3
IO
$\overline{IO}$
A
B
C
$\overline{G2A}$
$\overline{G2B}$
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
G1
$\overline{IS0}$
$\overline{IS1}$
$\overline{IS2}$
$\overline{IS3}$
$\overline{IS4}$
$\overline{IS5}$
$\overline{IS6}$
$\overline{IS7}$
+5
.01μ
.01μ
10μ
3.3 K
AK
BR
BS
IBW
$\overline{RD}$
RD
BP
$\overline{RD0}$
$\overline{IS0}$
$\overline{IS1}$
$\overline{RD1}$
BK
3.3 K
BJ
A12
A13
A14
A15
AW
AV
BN
AJ
IBA
8T26A
AY
D0
D1
D2
D3
BUS1
BUS2
BUS3
BUS4
$\overline{REC}$
XMIT
IN1
OUT1
IN2
OUT2
IN3
OUT3
IN4
OUT4
ID0
ID1
ID2
ID3
IBW
8T26A
AX
D4
D5
D6
D7
ID4
ID5
ID6
ID7
IBW

Figure 60 Application Module Interface Board - Address and Data Buffers

H. Motor Interface Module

The motor interface module (figures 60 and 61) consists of the logic required for the microcomputer to control the duty cycle of the SCR circuit; read the digitized armature current; time the interval between optical tachometer

IC Position	IC Type	Power	Gnd	Misc Connections
AJ	74LS266	14	7	
AK	3.3K Rs	14	-	
AL	74LS74	14	7	
AM	75492	11	4	
AN	DIP C.C.	-	-	
AP	74LS123	16	8	
AS	74LS138	16	8	
AT	8T98	16	8	gnd: 1,15
AV	8T98	16	8	gnd: 1,15
AW	8T98	16	8	gnd: 1,15
AX	8T26A	16	8	
AY	8T26A	16	8	
BJ	switches	-	-	gnd: 13 to 16
BK	3.3K Rs	14	-	
BL	74C14	14	7	
BM	74LS74	14	7	
BN	74LS04	14	7	
BP	74LS32	14	7	
BR	74LS32	14	7	
BS	74LS02	14	7	
BT	74LS20	14	7	
BV	74LS14	14	7	
BW	74LS368	16	8	
BW	74LS368	16	8	
BY	74LS368	16	8	
BZ	8253	24	12	
CN	2 of 3N28	-	-	
CP	2 of 3N28	-	-	
CR	2 of 3N28	-	-	
CS	2 of 3N28	-	-	
CT	2 of 3N28	-	-	
CV	2 of 3N28	-	-	
CW	2.2K Rs	14	-	
CX	2 of 3N28	-	-	
CY	3N28	-	-	
DL	8212	24	12	

Figure 61   Application Module Interface ICs

pulses; and detect timer overrun conditions. This logic appears to the microcomputer bus as a series of I/O ports. Interface between the outputs of this card and the external circuitry is provided using optical isolators; both for the protection of the microcomputer system, and to provide voltage translation where required.

The data bus is buffered by ICs AX and AY to provide eight bidirectional data signals, ID0 to ID7, for use on the interface card. The four most significant I/O address lines are compared to switch inputs by IC AJ, providing a board address valid signal (IBA) when a proper I/O address is output on the system bus. If the board address and the IOM signals are of the proper logic level, the data bus drivers are enabled in the bus write (CPU read) mode for the duration of the read (RD) pulse. At all other times the data bus buffers are active in the bus read (CPU write) mode. The three least significant address bits are decoded by IC AS to provide eight chip select outputs; enabled if address bit A3 is a logical 0.

The Front Panel LED Array

Eight light emitting diodes, located on the front panel, are enabled by IC DL located on the application module interface card (see figure 62). Since the LEDs are turned on when IC DL acts as a current sink, a logical 0 will turn the corresponding LED ON; while a logical 1 will turn it OFF. The one ohm resistor and the 0.1 microfarad capacitor, through which the LED current is sourced, serve to decouple the LED switching surges from the system power bus. The output port is selected by an active write signal ANDed with chip select six, resulting in a port address of

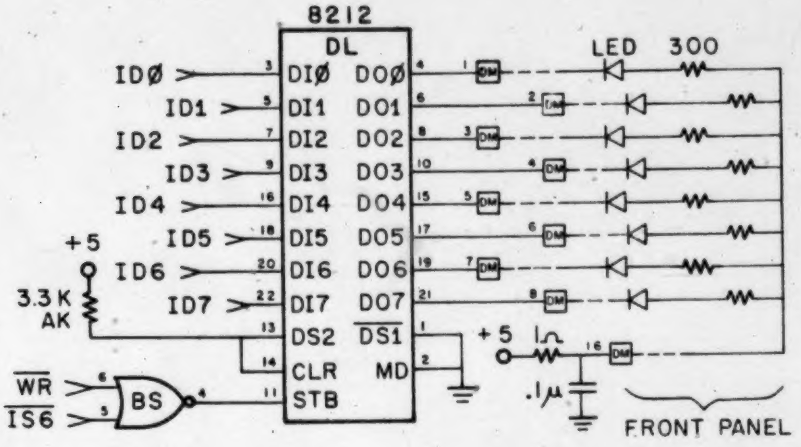

8212
DL
ID0
ID1
ID2
ID3
ID4
ID5
ID6
ID7
DI0
DI1
DI2
DI3
DI4
DI5
DI6
DI7
DS2
CLR
STB
DO0
DO1
DO2
DO3
DO4
DO5
DO6
DO7
DS1
MD
+5
3.3 K
AK
WR
IS6
BS
LED
300
+5
1Ω
.1μ
FRONT PANEL

Figure 62   Front Panel LED Array

nnnn 0110 (binary); where nnnn is selected by the board address switches.

I. The Programmable SCR Circuit

Motor control is achieved by varying the average DC output voltage of the SCR/rectifier bridge (see figure 63). This average output voltage is varied by introducing a delay between the AC voltage zero and the leading edge of the SCR firing pulse. An MC3370 zero voltage crossing detector IC, located in the application module, produces a current pulse of 100 microsecond duration, centered about each AC line voltage zero. This current pulse is transmitted, via an optical isolator (IC CN) and an inverter with a Schmitt trigger input (IC BV) to the gate of timer 0 of the 8253. At this point, the rising edge of the ZCD pulse precedes the actual voltage zero by 40 microseconds. The rising edge of

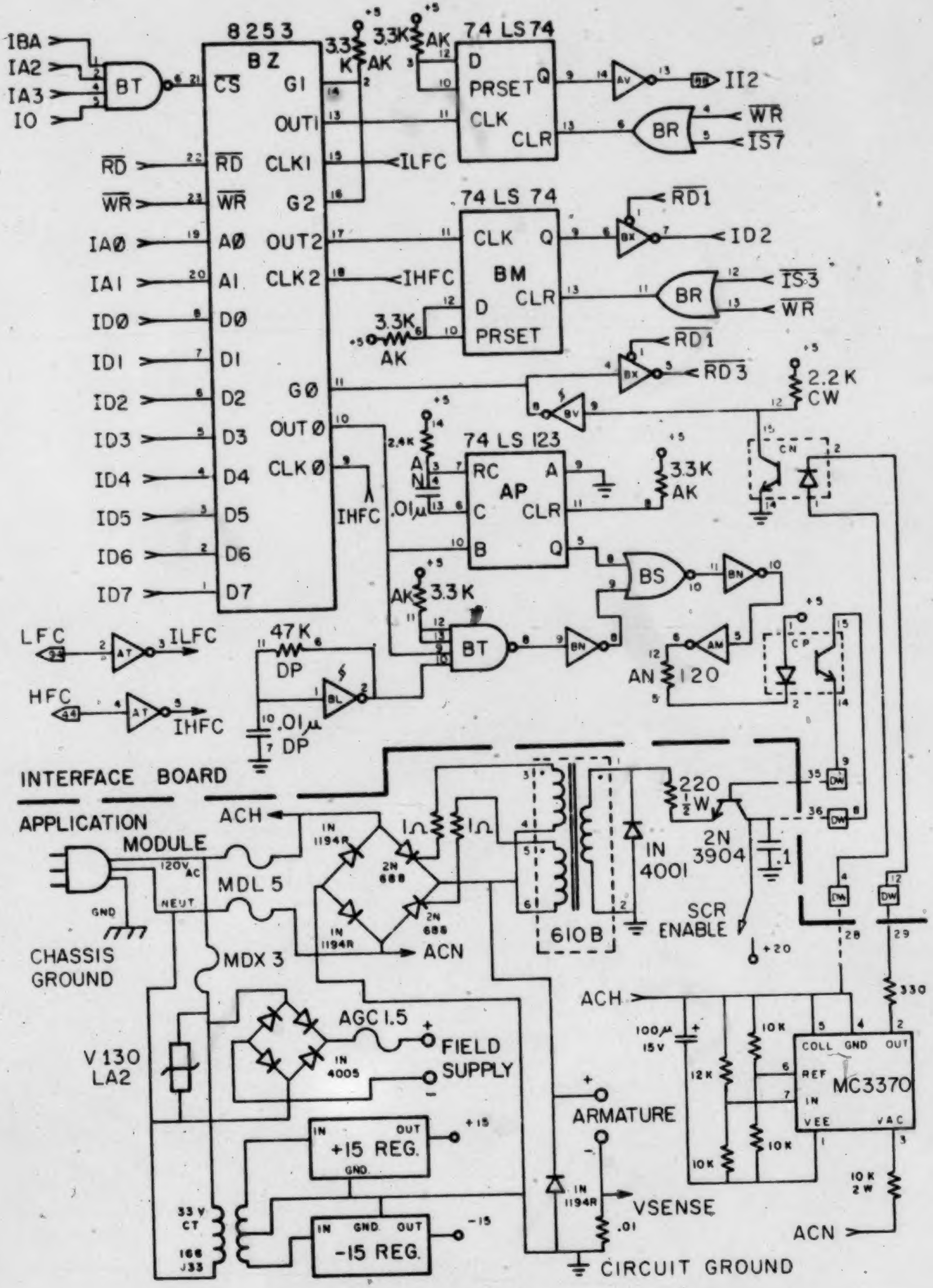

8253
BZ
IBA
IA2
IA3
IO
BT
CS
G1
OUT1
CLK1
G2
OUT2
CLK2
RD
WR
IA0
IA1
A0
A1
ID0
ID1
ID2
ID3
ID4
ID5
ID6
ID7
D0
D1
D2
D3
D4
D5
D6
D7
G0
OUT0
CLK0
IHFC
ILFC
3.3K AK
+5
74 LS74
D
PRSET
Q
CLK
CLR
AV
II2
BR
WR
IS7
74 LS 74
BM
RD1
BX
ID2
IS3
RD3
BV
2.2K CW
CN
74 LS 123
AP
RC
A
C
B
2.4K
AN
.01µ
BS
BN
AK 3.3K
AM
CP
AN 120
LFC
HFC
AT
47K DP
BL
.01µ DP
INTERFACE BOARD
APPLICATION MODULE
ACH
ACN
120V AC
NEUT
GND
CHASSIS GROUND
MDL 5
MDX 3
IN 1194R
2N 688
1Ω
610B
IN 4001
220 ½W
2N 3904
.1
SCR ENABLE
+20
DW
V130 LA2
AGC 1.5
IN 4005
FIELD SUPPLY
ARMATURE
+15 REG.
-15 REG.
IN
OUT
GND.
+15
-15
33 V CT
166 J33
VSENSE
.01
CIRCUIT GROUND
100µ 15V
12K
10K
330
10K 2W
MC3370
COLL
GND
OUT
REF
IN
VEE
VAC

Figure 63 Motor Control Circuitry

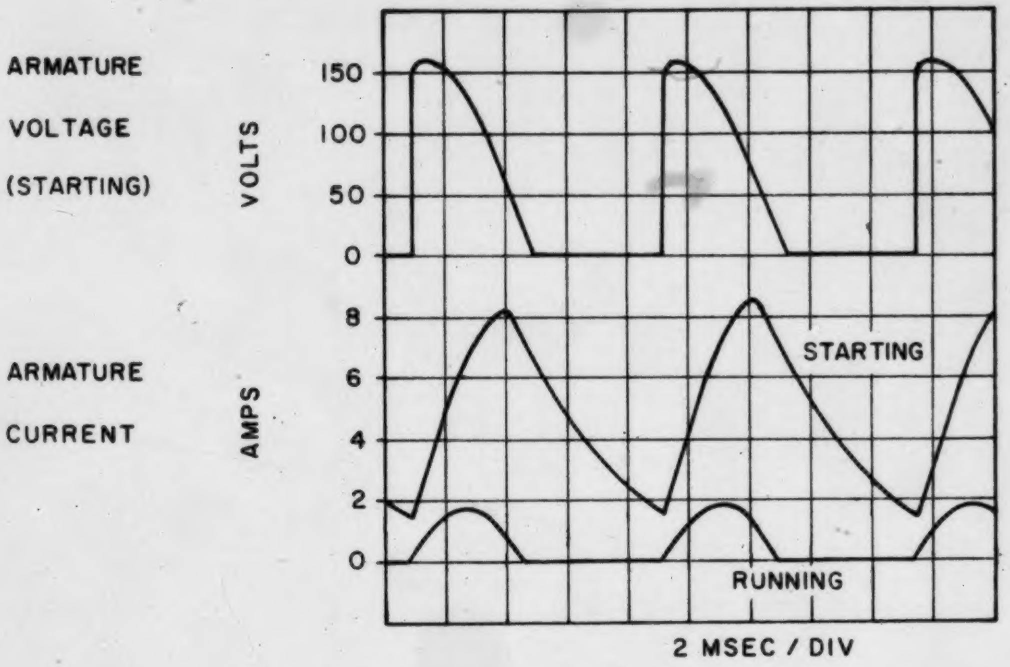


Figure 64 Armature Voltage and Current - 50% Duty Cycle

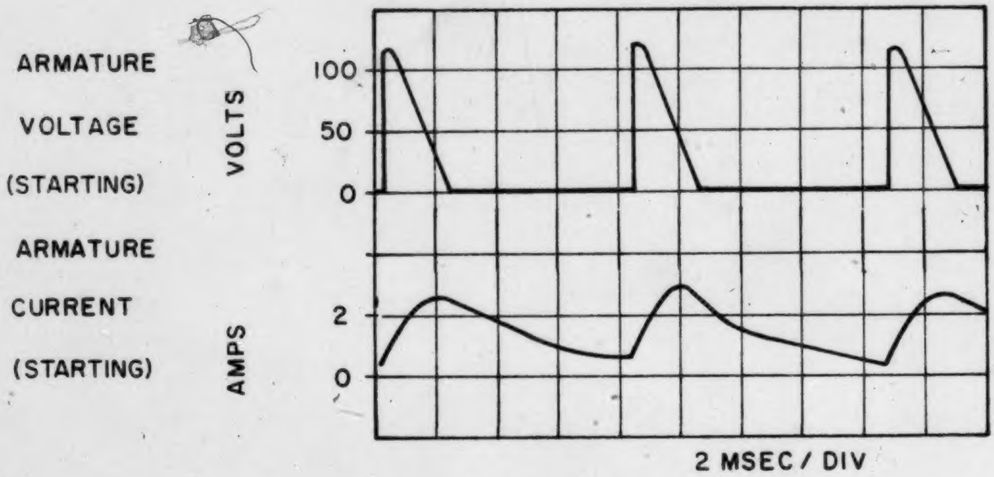


Figure 65 Armature Voltage and Current - 25% Duty Cycle

the ZCD pulse causes the output of timer zero to fall to the logical 0 level; disabling the SCR circuit. The timer then counts the incoming 1 MHz clock pulses, counting down from the program stored terminal count, until zero is reached. At this point the counter output rises to the logical 1 level. This rising edge is detected by a one shot, IC AP, which outputs a pulse of ten microsecond duration. This pulse is transmitted to the application module through optical isolator CP. The optical isolator output transistor forms a darlington pair with a 2N3904, allowing a current pulse to flow through the Hammond 610B pulse transformer. The secondaries of the pulse transformer provide gate current to two 2N688 SCRs, allowing the one that is presently forward biased to turn on.

If at this point in time both SCRs are reverse biased due to the fact that the armature back EMF exceeds the line voltage, neither SCR will turn on; resulting in a lost power cycle. IC BL, a CMOS Schmitt trigger input inverter, together with its related resistor and capacitor, forms a 50 KHz square wave oscillator. These output pulses are ANDed with the SCR enable signal by IC BT. The resulting pulse train is ORed with the SCR firing pulse originating with the one shot (IC AP) by IC BS. Thus if the SCR is not fired by the original trigger pulse, it will be activated by the oscillator output as soon as the line voltage exceeds the motor back EMF.

A switch has been placed in series with the unregulated

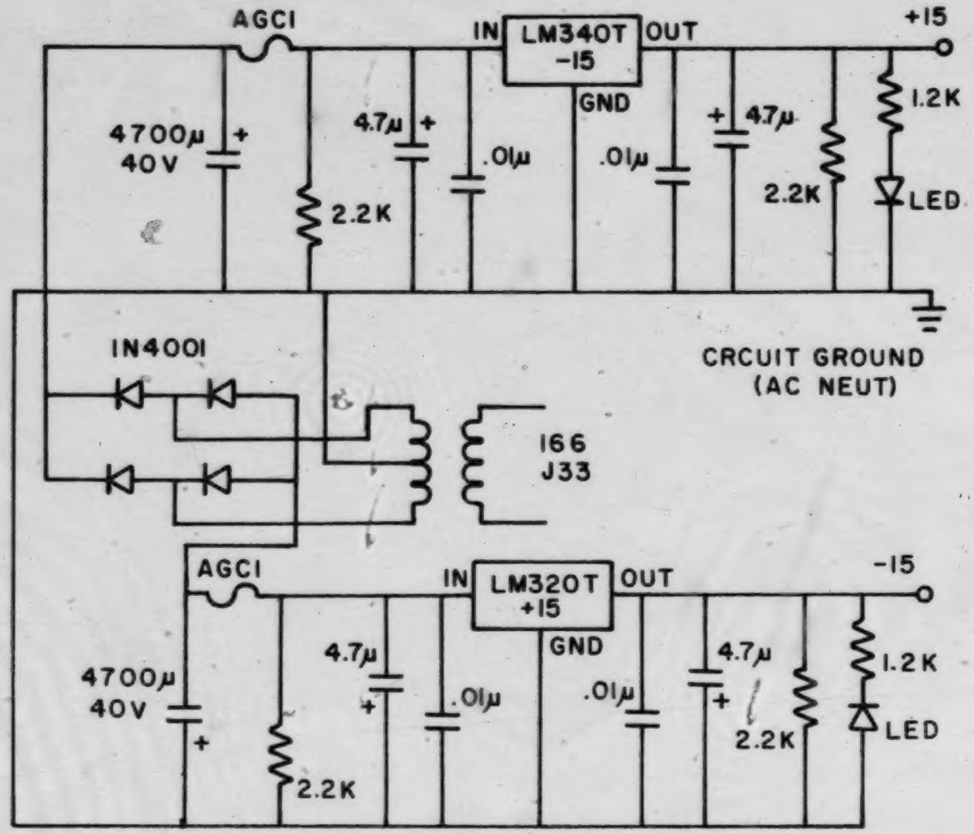

AGCI
IN
LM340T
-15
OUT
+15
GND
4700µ
40V
+
4.7µ
.01µ
2.2K
1.2K
LED
CRCUIT GROUND
(AC NEUT)
IN4001
166
J33
AGCI
IN
LM320T
+15
OUT
-15
GND
4700µ
40V
4.7µ
.01µ
2.2K
1.2K
LED

Figure 66 Application Module Power Supply

+15 volt supply (nominal +20 volts) feeding the SCR firing circuit, to allow the system user to quickly deactivate the SCR system, or to leave it disabled during program debugging.

Figures 64 and 65 illustrate typical voltage and current waveforms obtained when the variable duty cycle SCR/diode bridge is used to drive a DC motor armature. The starting voltage and current waveforms correspond to a motor near standstill, thus the armature back EMF is negligible, allowing the armature voltage to drop to zero when the SCRs are disabled. The low back EMF also allows maximum armature current to flow. Due to the inductance of the armature winding, the armature current continues to flow through the free wheeling diode after armature voltage returns to zero. Once the motor RPM increases, the effective armature voltage is significantly reduced by the back EMF, resulting in a considerably reduced armature running current.

The Motor Field Supply

A diode bridge, consisting of 1N4005 diodes, has been provided to rectify the 120 V AC to provide a motor field supply. A 1.5 amp fuse is used to protect these diodes.

Application Module Power Supplies

Plus and minus 15 volt supplies (see figure 66) provide the regulated DC voltage required by the application module's analog and digital circuitry.

The analog circuit ground in the application module is tied to the AC neutral side of the current to voltage transducer. Thus the ground potential is a diode forward voltage drop above the AC line neutral potential. This ground reference permits the filtering and conversion of the analog signal without having to compensate for a large common mode input voltage.

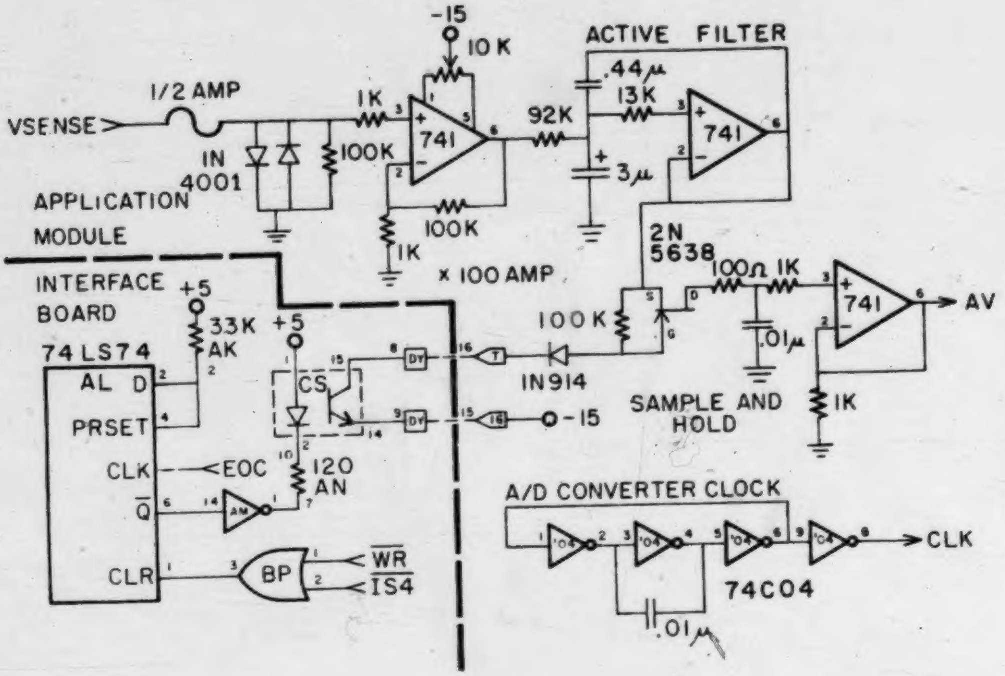

-15
10 K
ACTIVE FILTER
.44µ
13K
1/2 AMP
1K
VSENSE
741
92K
741
IN 4001
100K
3µ
APPLICATION
MODULE
100K
1K
× 100 AMP
2N 5638
100Ω
1K
INTERFACE
BOARD
+5
33K
AK
+5
100 K
741
AV
74 LS74
AL
D
PRSET
CLK
Q̄
CLR
CS
DY
T
IN914
DY
16
-15
.01µ
SAMPLE AND HOLD
1K
EOC
120
AN
AN
A/D CONVERTER CLOCK
'04
'04
'04
'04
CLK
BP
WR
IS4
74C04
.01µ

Figure 67 Armature Current Sense Amplifier

J. Armature Current Sensor

The armature current transducer consists of a sensing resistor in series with the armature. The voltage across this resistor is amplified, filtered, and latched in a sample and hold circuit. The latched analog voltage is converted to an 8 bit binary number, which is read by the microcomputer.

The Current Sense Amplifier and Filter

Figure 67 illustrates the circuitry used to amplify and filter the signal from the armature current transducer. The armature current is converted to a voltage using a meter shunt specified to produce a 50 mV signal if a five amp current is passed through it. The 100% overload rating allows it to be used for up to 10 amps. Input voltage spikes to the amplifier circuit are clipped at plus and minus 700 mV by the two 1N4001 diodes. Should the shunt circuit accidentally develop an open circuit; the application of full 120 volt DC to the input of the operational amplifier is prevented by these diodes. The diodes will limit this input voltage to 700 mV until the half amp fuse opens, isolating the analog circuit from the voltage source. The 100 Kohm resistor on the operational amplifier input provides a ground reference for zero armature current.

The amplifier has a gain of 100, thus its output voltage corresponds to the armature current in amps, with amplifier saturation taking place at about 13 volts peak (corresponding to 13 amps peak armature current). The zero offset voltage for the entire analog circuit is trimmed out using the 10 Kohm potentiometer connected to the amplifier IC. A scaling factor in software is used to trim the amplifier gain. Sample current waveforms obtained at the output of the amplifier are illustrated in figure 64 and 65.

Before conversion to a binary number, the rapidly

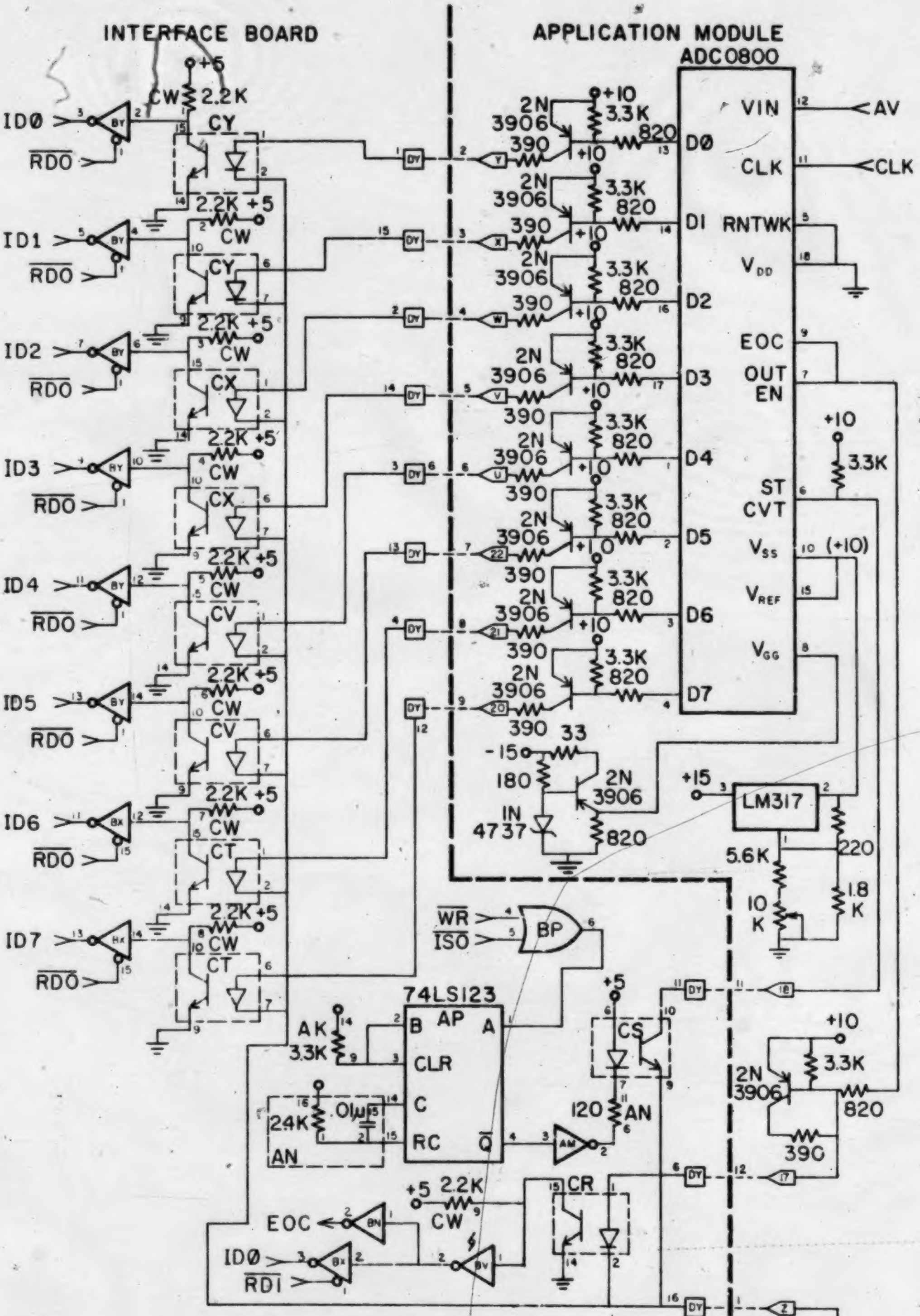

INTERFACE BOARD
APPLICATION MODULE
ADC0800
ID0
ID1
ID2
ID3
ID4
ID5
ID6
ID7
RDO
+5
2.2K
CW
CY
CX
CV
CT
BY
BX
DY
2N
3906
390
3.3K
820
+10
D0
D1
D2
D3
D4
D5
D6
D7
VIN
AV
CLK
RNTWK
V_DD
EOC
OUT
EN
ST
CVT
V_SS
(+10)
V_REF
V_GG
-15
33
180
IN
4737
+15
LM317
220
5.6K
10
K
1.8
K
WR
ISO
BP
74LS123
AP
A
B
CLR
C
RC
Q
AK
3.3K
24K
.01μ
AN
120
CS
CR
BN
BV
EOC
ID0
RD1

Figure 68   Analog to Digital Converter Interface

varying amplifier output must be converted to a DC level proportional to the average value of the armature current. This averaging is done using an active, low pass, unity gain filter [42] with a rise time of one second. The resulting expected maximum variation over one half of a 60 Hz cycle is estimated to be 60 mV; compared to an A/D converter resolution of 40 mV. The maximum observed variation over half of a 60 Hz cycle is in the order of 30 mV. The delay introduced by this filter has the added advantage of smoothing over very short high current pulses, preventing unnecessary system shut down by the digital overcurrent monitoring program.

The Sample and Hold Circuit

The output of the averaging filter is fed to a storage capacitor through a junction FET. Under normal conditions, the FET is in the low resistance state, allowing the capacitor voltage to track the filter output. A write instruction to the proper I/O port causes $\bar{Q}$ of IC AL (figure 67) to become a logical 1, causing current to be drawn from the gate of the FET; putting the channel into a high impedance mode. The voltage stored in the hold capacitor, buffered by a unity gain amplifier, is then converted by the A/D converter. The end of conversion pulse, EOC, toggles a flip flop when the analog to digital conversion is completed; returning the FET to a low impedance state so that the hold capacitor voltage can continue to track the filter output. A 100 ohm resistor limits the capacitor charging current flowing through the FET; whereas the 1 Kohm resistor protects the unity gain amplifier input from a current surge during power shutdown.

Analog to Digital Converter Circuitry

The analog to digital converter (see figure 68) is operated with Vss = 10 volts; in order to obtain an input range of 0 to 10 volts; rather than a -5 to +5 volt input range. A bias supply and the 10 volt reference is generated from the +15 volt supply using an LM317. A one shot, IC AP, is used to stretch the start conversion pulse from IC BP into a pulse of 10 microsecond duration. This pulse is then coupled to the A/D converter through optical isolator IC CS. The A/D converter's end of conversion output is used to enable the A/D converter data outputs, as well as to return the sample and hold circuit to the sample mode. The EOC signal is also available to the CPU so that the completion of the conversion can be determined. The A/D converter data outputs are coupled to the interface board data bus through optical isolators and tristate buffers.

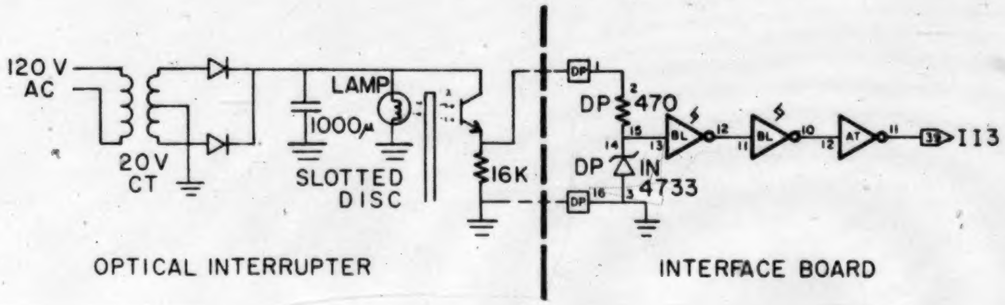

120 V
AC
20 V
CT
1000μ
LAMP
SLOTTED
DISC
16K
DP
DP
470
DP
IN
4733
DP
BL
BL
AT
39
I13
OPTICAL INTERRUPTER
INTERFACE BOARD

Figure 69 Optical Tachometer System

K. Optical Tachometer

The optical tachometer system (see figure 69) provides a system interrupt every 45 degrees of motor shaft rotation. The output of the existing optical interrupter is normally pulled to ground by the 16 Kohm resistor. When the phototransistor receives a light pulse via a slot in the disc, the transistor conducts, pulling the output high. The resistor, zener diode and Schmitt trigger inverter (IC BL) shape this output pulse. IC AT then buffers this signal onto the II3 line, activating the edge triggered interrupt, RST 7.5, of the 8085. This interrupt is internally latched until it is serviced.

VI. Software Documentation

The system software is divided into four functional sections: the keyboard monitor, the floating point arithmetic package, a set of routines to allow the microcomputer system to be used as a calculator, and an application module consisting of a DC motor speed control program. Figures 70 to 72 provide a general breakdown of memory and I/O usage.

Programming Conventions

Several conventions have been adopted and adhered to in order to produce a software package that is easy to understand and modify, yet is flexible enough to be of maximum use.

To simplify debugging and modification, major programs have been broken down into functional blocks. Each of these blocks performs a well defined operation by itself; and being a subroutine, is capable of being debugged on its own. A major program thus consists in a large part of a series of pretested subroutines, greatly simplifying software modification and debugging.

Since software will be stored in 1024 byte blocks of read only memory, the source code should be structured with this maximum block size in mind [40]. Thus, the program is written in blocks, each of which will fit into a single 2708 read only memory chip. Each program or subroutine that is

Read Only Memory IC	Memory Page (Decimal)	Memory Address (Hexadecimal)	Contents
0	0 - 3	0000 - 03FF	Keyboard Monitor
1	4 - 7	0400 - 07FF	Floating Point Addition, Subtraction, Multiplication and Division
2	8 - 11	0800 - 0BFF	Decimal to Binary and Binary to Decimal Conversion
3	12 - 15	0C00 - 0FFF	Calculator Functions
4 - 5	16 - 23	1000 - 17FF	Motor Control Software
6 - 7	24 - 31	1800 - 1FFF	Unused

Figure 70 Memory Map - Read Only Memory

Random Access Memory Bank	Memory Page (Decimal)	Memory Address (Hexadecimal)	Contents
0	32	2000 - 2057	Monitor Program Data
0	32	2058 - 2066	Application Program Data
0	32 - 33	2067 - 21FF	Unused
0	34 - 35	2200 - 23FF	Reserved for Stack
1	36 - 39	2400 - 27FF	User Workspace

Figure 71 Memory Map - Random Access Memory

referred to from outside its block has a 'forwarding address' located at a fixed location at the beginning of the block in which it resides. Any external reference is then made to the fixed location of this 'forwarding address', rather than to the actual starting address of the routine. As a result, when a routine is modified, addresses within the program block itself are readjusted, but the location of the forwarding address remains fixed. Thus when a change is made, only the IC directly affected needs to be reprogrammed; all external addresses referenced to the altered memory remain fixed.

A. The Keyboard Monitor

The keyboard monitor, located in the first four pages of memory, provides the operator with a means of controlling the activities of the computer system. The heart of the monitor consists of a routine which determines which keyboard switch was closed, and then calls the appropriate subroutine to execute the desired function. These monitor functions include routines which enable the user to read from or write to memory locations or I/O ports, or to initiate program execution at any desired address.

I/O Port Address (Hexadecimal)	Label	I/O Port
04	D8279	Display Data Register
05	C8279	Keyboard/Display Control Register
F0	ADC01	A/D Converter Data Output
F1	FFDAT	Flip Flop Data
F3	TRS12	RPM Timer Overrun Reset
F4	LATSH	Sample/Hold Gate Latch
F6	FPLED	Front Panel LED Array
F7	TRS11	Interrupt Reset; Control Update Timer
FC	TIM10	Timer IC 1, Counter 0
FD	TIM11	Timer IC 1, Counter 1
FE	TIM12	Timer IC 1, Counter 2
FF	TIM1C	Timer IC 1, Control Register

Figure 72 Memory Map - I/O Ports

Interrupt Vectors

A signal applied to one of the interrupt lines of the 8085 will result in the processor fetching an instruction from a predetermined memory location. Jump instructions are stored at these locations which then pass system control to the routine required to service the interrupting device.

The reset signal will cause program control to be transferred to the reset routine, which initializes the microcomputer system. The internal interrupt, RST 5.5, initiates the execution of the keyboard service routine. RST 7.5, an edge triggered interrupt originating in the optical tachometer circuit, requests the execution of the motor control program. The remaining interrupt vectors, if accessed, will result in the execution of an error exit.

Error Exits

Throughout the software, abnormal conditions are tested for, and if found will result in an error exit. If this occurs, the internal registers affected by the exit routine are stored; an error code in seven segment form is loaded into the HL registers, and a jump is executed to the exit routine via a 'forwarding address'.

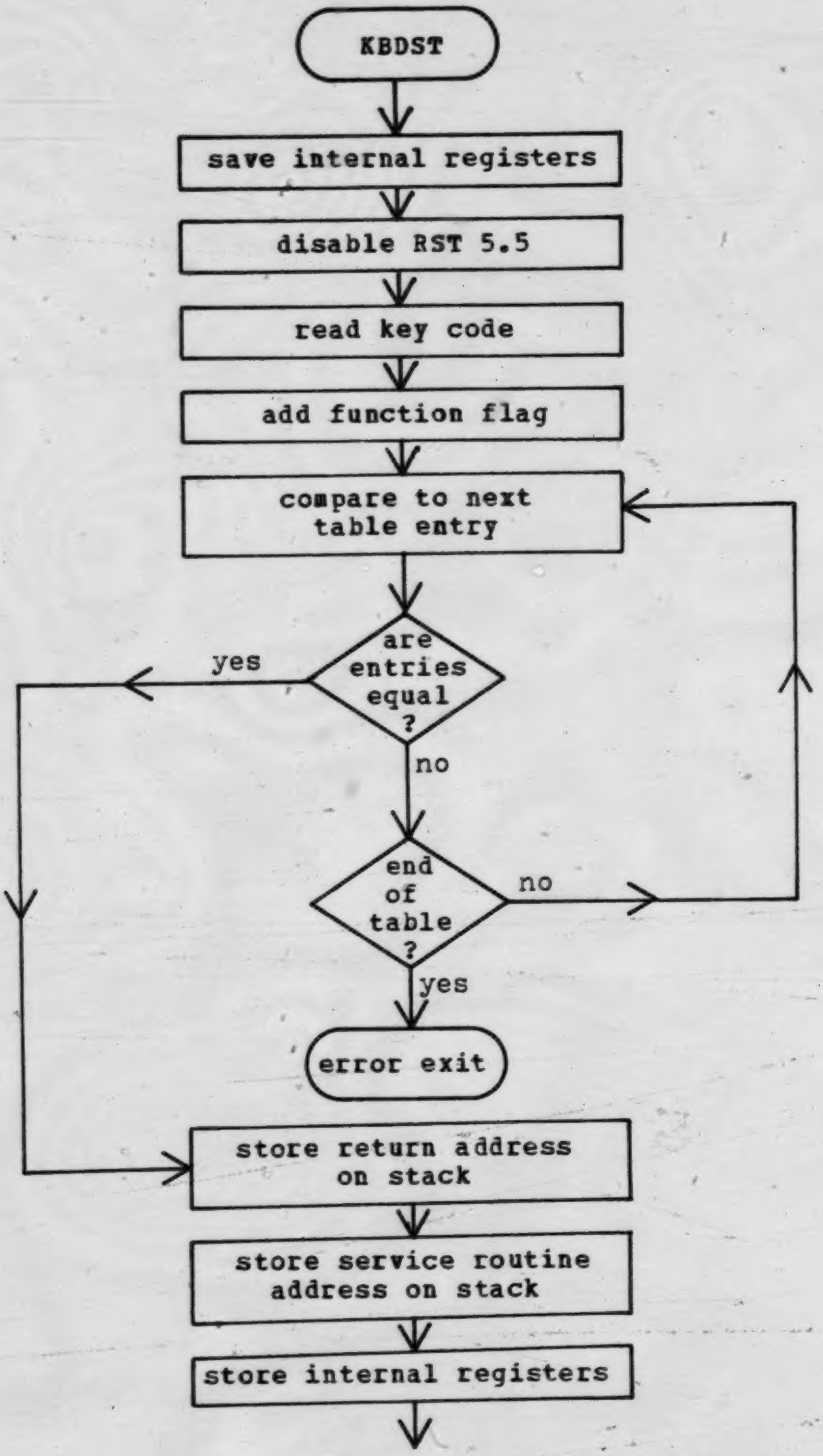
In the exit routine itself, the error code is displayed and the internal CPU registers are restored to the state in which they were when the error was detected. A halt instruction is then executed.

System Reset

The function of the reset routine is to initiate the operation of the microcomputer system in an orderly manner. The location of the top of the stack is defined to be 23FF hex, and the operating mode of the 8279 keyboard/display interface is defined. The FIFO register and the display memory in the 8279 are then cleared. Since this leaves the display RAM inaccessible during the clearing operation, the CPU waits until it is completed. The display clear subroutine is then called to clear various flags. The interrupt system is initialized with only RST 5.5 (the keyboard interrupt) active, and the 'forwarding addresses' associated with the auxiliary functions are defaulted to an error exit. Program control is then transferred to an idle loop to await further instructions. Note that at this point the SOD line is reset. In general, when the system is executing an interrupt initiated function, the SOD line is set; otherwise it is reset. This allows the system user to determine when the microcomputer is active and how long is required to perform certain functions, by simply observing the SOD output on an ocilloscope.

The Keyboard Monitor Routine

The keyboard service routine, flowcharted in figure 73, enables program control to be transferred to a keyboard selected subroutine. An interrupt on the RST 5.5 line results in a subroutine call of the address KBDST. The CPU's internal registers are then pushed onto the stack; the RST 5.5 interrupt is disabled to prevent another interrupt before the present one is serviced; and the data byte corresponding to the pushed key is read from the 8279. An offset is added to this byte to indicate whether or not the function flag is set; resulting in the function code. The function code is a unique data byte, corresponding to the desired function. This data byte is compared to the first byte of each three byte record stored in a data file. If a


KBDST
save internal registers
disable RST 5.5
read key code
add function flag
compare to next table entry
are entries equal ?
yes
no
end of table ?
no
yes
error exit
store return address on stack
store service routine address on stack
store internal registers

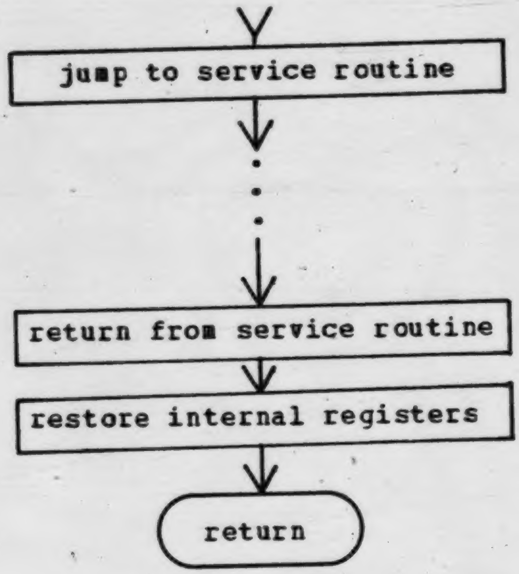


Figure 73   Keyboard Service Routine Flowchart

match is obtained, the second two bytes of the record, containing the address of the desired service routine, are pushed onto the stack, along with the address of the return routine. The microprocessor's internal registers are then restored to their preinterrupt condition, and a subroutine call to the desired keyboard service routine is simulated. The subroutine's return statement will transfer control to the return routine, where the RST 5.5 interrupt is enabled for the next key operation. The microprocessor's internal registers are returned to the preinterrupt condition and program control is returned to the routine inerrupted by the key operation.

Two error exits may be encountered in the execution of this program. Error code 2 is the result of pushing a keyboard element for which no service routine has been defined. This error may occur, since keys have been left unassigned for future applications. Error code 3 indicates that the end of the lookup file was reached before an entry equal to the function code was found.

The Display Clear Routine

This keyboard callable routine clears the display and the corresponding memory file, as well as setting the pointer to the leftmost display digit. The function flag (FTNFL), data flag (DATAF), clear flag (CLRFL), and the display flag (DISPF) are also reset. Internal microprocessor registers are not affected by this routine.

Data Entry Routines

Pressing a data key (0 to F) will cause the digit to appear in the display, while its hexadecimal value is stored in the corresponding memory file location. Each key has a unique entry point (labels BUT00 to BUT0F), where the hexadecimal digit and its seven segment code are entered into an internal microprocessor register. From there the program jumps to DSPST, the beginning of the program common to all data keys.

The clear flag is tested, and if set, the display is cleared by calling the display clear routine. The display digit counter is then loaded (DISCT), and used as an offset to store the required digit in the proper display and memory file location. The display digit counter is then decremented and tested. If the new display digit is positive it is stored in memory, otherwise the data/address flag (DATAF) is loaded. If the data/address flag is set, the display digit counter is reset to 7, otherwise it is reset to 1 for data byte modification. The new display digit counter is then stored in memory.

Load Address Routine

This keyboard callable routine loads a four digit hexdecimal address, packs it into two bytes, and stores it in the HL registers of the microprocessor. The memory display routine is then called to display the contents of the addressed memory byte, and store the address itself in a preassigned memory location. Upon return to the load address routine, the data flag (DATAF) is set to indicate that the next entry will be data. The display digit counter (DISCT) is set to one so that entered data will be loaded into the data field.

Reading and Displaying a Memory Location

This routine loads the contents of the memory location, addressed by the HL pair, into the B register. The address is then stored in a preassigned memory location, starting at ADDRL, for later use. The data byte, now in the B register, is unpacked into two hex digits, which are converted to their seven segment code and output to the two rightmost display locations. The two address bytes are also displayed, followed by a dot to separate them from the data byte. Conversion to seven segment code is done using the hex to seven segment conversion routine.

The Memory Write Function

The memory write function, a keyboard service routine, loads the two hex digits just entered, packs them into a single data byte, and stores this byte in the memory location whose address is stored in ADDRL and ADDRH (least significant byte and most significant byte respectively). The addressed memory location is then read to insure that the data byte was properly stored. If the read data differs from the written data, an error message is displayed. If the proper storage of the data byte is confirmed, the address now stored in the HL pair is incremented, and the memory display subroutine is called to store the incremented address in memory, and display it along with the data byte presently stored at that location.

The Memory Read Function

This routine services the memory read key. The data address, stored starting at ADDRL, is loaded into the HL registers and incremented. The memory display routine is then called to store the new address, and display it and the data byte presently stored there.

The Run Function

Depressing the RUN key causes program execution to begin at the previously entered address. The stack pointer is rolled back, to permit a subroutine return from the program accessed via this function to return to the keyboard return routine. RST 5.5 is then enabled and SOD is cleared. Program control is now transferred by loading the desired address from the HL register pair to the program counter.

Setting the Function Flag

The function key (FTN) is serviced by this routine. A hexadecimal number, 80, is loaded into the A register. This number is then stored in the function flag (FTNFL) location, to be later used as an offset by the keyboard service routine.

The I/O Port Read Function

This routine services the I/O port read key (IO RD). The function flag is cleared and the two hexadecimal digits defining the port address are loaded and packed into a single byte. A subroutine to read the desired port is then written into RAM and executed, resulting in the contents of the desired port being loaded into the microprocessor's A register. The port data display routine is then called to display the port address and its contents.

Port Data Display

This routine (starting label IDSST) is entered with the port number in the B register, and its contents in the C register. A 'P' is displayed, followed by the port address and a dot. This is then followed by the data byte. The hexadecimal to seven segment conversion routine is called upon for the required conversions.

The I/O Port Write Function

The I/O port write key (IO WR) requests the execution of this routine. The two address nibbles (a nibble consists of four bits) are loaded into internal microprocessor registers from the memory file corresponding to the displayed data. These two nibbles are then packed into a single byte. Similarly, the data nibbles are then loaded and packed into a single byte. A subroutine to write the data to the required port address is then loaded into RAM and executed. The port data display routine is then called to display the port address and the data written to the port.

Hexadecimal to Seven Segment Code Conversion

This routine converts the hexadecimal digit stored in the A register into the corresponding seven segment code. The hexadecimal digit is added to a base address to access a table entry, which contains the required seven segment code.

The Idle Loop

While awaiting further instructions, the microcomputer executes an endless idle loop. The serial data output is toggled during the execution of this loop, enabling a quick oscilloscope check for hardware failure of the basic microcomputer system.

The Data Block Move Routine

This routine moves a data block from one starting address to an other. The source starting address is stored in the BC register pair; the destination starting address is in the DE pair; and the number of bytes to be moved is in the H register.

B. The Floating Point Arithmetic Package

The floating point arithmetic package, residing in memory pages 4 to 11, contains the necessary software to perform binary floating point multiplication, division, addition and subtraction, as well as conversion between floating point binary and decimal numbers [43-44].

Binary Floating Point Number Format

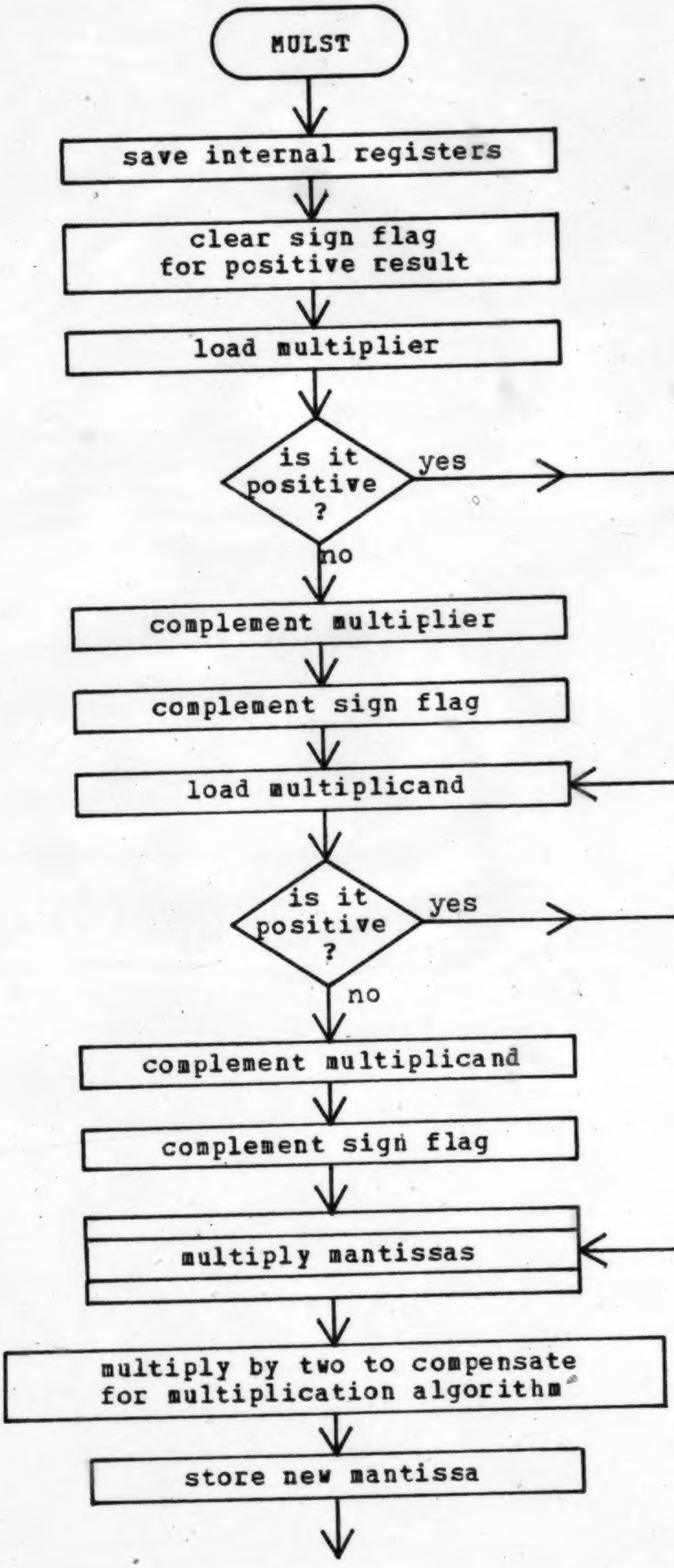
All floating point binary numbers are expressed in the following three byte format:

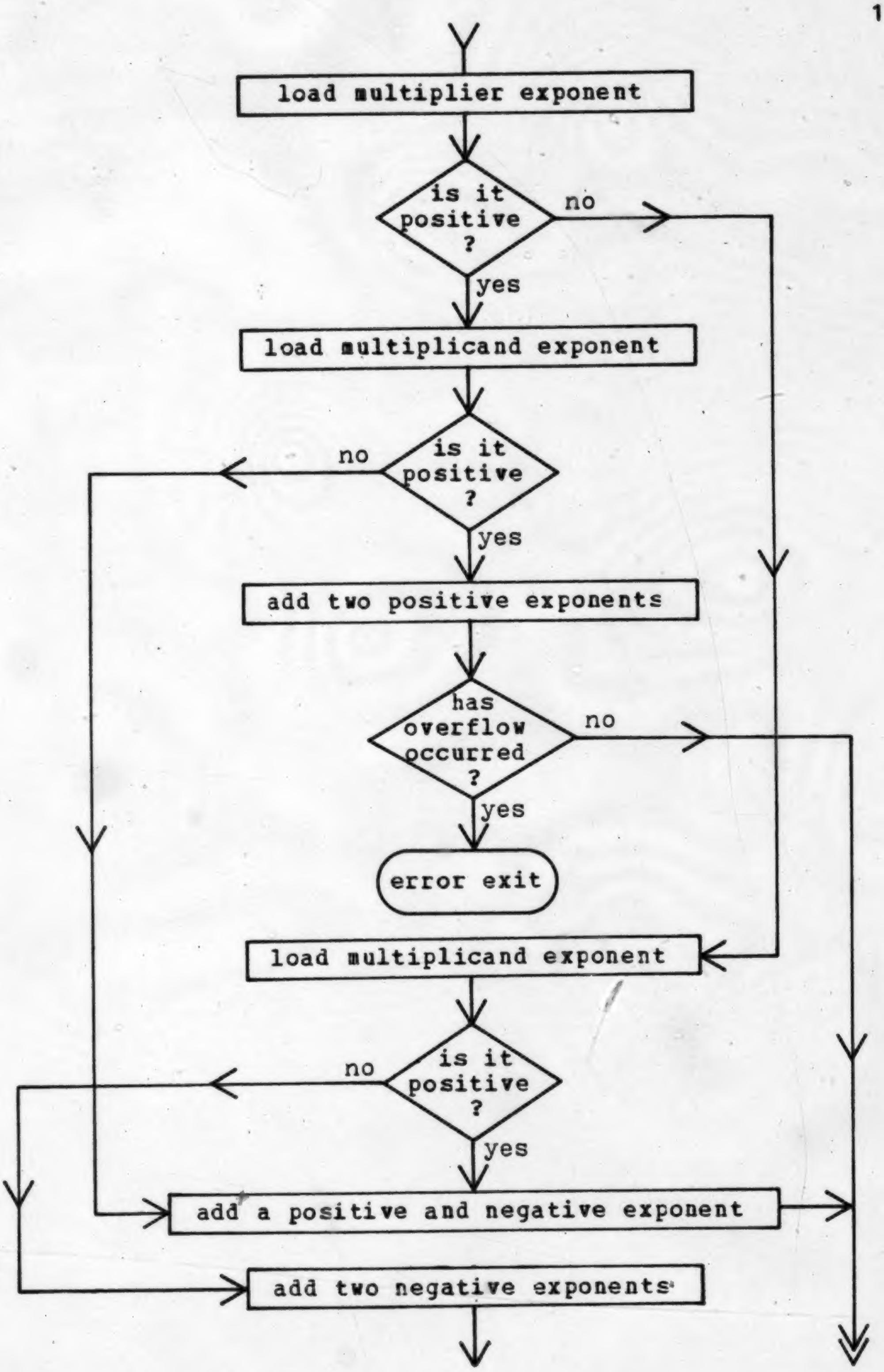
S.MMM MMMM MMMM MMMM SEEE EEEE

The mantissa consists of a sign bit (S) and a 15 bit mantissa (M), expressed in two's complement form. The exponent is adjusted such that the magnitude of the mantissa is always less than 1.0, but larger than or equal to 0.5 (decimal). The exponent consists of a sign bit followed by 7 exponent (E) bits, again expressed in two's complement form. The base of the exponent is 2.

Floating Point Multiplication

The floating point multiplication routine loads the two binary floating point numbers to be multiplied from FPV1 and FPV2, and converts them to positive numbers for multiplication. The sign flag, SIGNF, is used to indicate the polarity of the resulting product. The 16 bit binary multiplication routine is then called to perform the actual multiplication. Due to the format's requirement of a zero


MULST
save internal registers
clear sign flag for positive result
load multiplier
is it positive ?
yes
no
complement multiplier
complement sign flag
load multiplicand
is it positive ?
yes
no
complement multiplicand
complement sign flag
multiply mantissas
multiply by two to compensate for multiplication algorithm
store new mantissa


load multiplier exponent
is it positive ?
no
yes
load multiplicand exponent
is it positive ?
no
yes
add two positive exponents
has overflow occurred ?
no
yes
error exit
load multiplicand exponent
is it positive ?
no
yes
add a positive and negative exponent
add two negative exponents

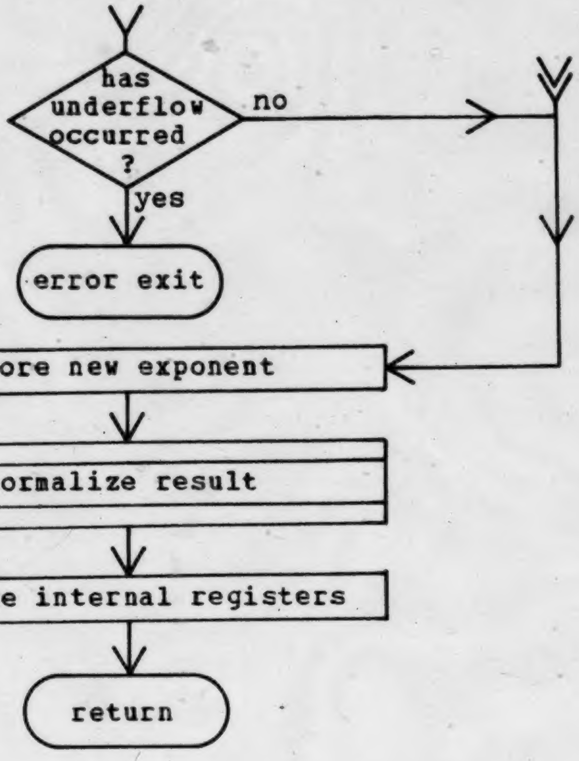

has underflow occurred ?
no
yes
error exit
store new exponent
normalize result
restore internal registers
return

Figure 74 The Floating Point Multiplication Routine

leading bit for each positive mantissa, a product will be obtained with two leading zeroes. To compensate for this, the product is shifted one bit to the left. The two exponents are then added, and tested for overflow or underflow. The normalization subroutine is then called to add the proper sign to the mantissa, and convert the result to the proper format. The result is stored in FPV1.

Sixteen Bit Binary Multiplication

This subroutine multiplies the 16 bit integer stored in the BC register pair by the 16 bit integer stored in the DE register pair. The multiplication is performed by sampling the multiplier, bit by bit, starting with the least significant bit. If the bit is a one, the 16 bit multiplicand is added to the most significant 16 bits of the product; no addition is performed if the current multiplier bit is a zero. The multiplier is then shifted right to test the next bit; while the product is shifted right one location due to the increased power of two represented by the next multiplier bit. This shifting, testing and adding is repeated 16 times, once for each multiplier bit. When completed, the multiplicand added (or not added) to the product due to the 2^0 bit of the multiplier, will have been shifted to the least significant bits of the product; while the addition due to the most significant multiplier bit will have its effect felt in the most significant product bits. The 32 bit product is then stored such that the most significant 16 bits are in the HL register pair, while the 16 least significant bits are in the DE register pair.

The Normalization Routine

The normalization routine converts the sign flag, mantissa, and exponent resulting from a previous operation into a valid binary floating point number. The incoming, positive, four byte mantissa is stored in SHFT1 to SHFT4 (most significant to least significant byte) with the decimal point located after the most significant bit of SHFT1. The two's complement exponent is stored in SHFT5, and the sign flag (set for -, reset for +) is stored in SIGNF.

Entry into the normalization routine is defined such that the incoming mantissa is always positive, and always less than one in magnitude. If a zero mantissa is detected, the proper floating point format for a zero is stored, completing the function of this routine. If the mantissa is not zero, it is shifted left until its magnitude is greater than or equal to 0.5 (decimal), and less than 1.0, as required, with the exponent adjusted accordingly. When of the proper magnitude, the mantissa is tested to determine if roundup is required. If it is required, the proper rounding operation is performed. The final exponent is then tested for underflow. If an error is detected, error code 6 is displayed, otherwise, the sign flag is loaded and tested. If it is set, the two's complement of the mantissa is taken. The resulting valid floating point number is then stored in FPV2, completing the normalization routine.

Floating Point Division

This routine divides a binary floating point dividend stored in FPV1 by a binary floating point divisor, stored in FPV2. The dividend and divisor mantissas are loaded and converted to positive quantities if required, with the sign flag adjusted to give the proper quotient sign. The divisor is tested, and if it is zero, an error exit is performed, displaying error code 8.

Division then proceeds in the following manner. The divisor is subtracted from the dividend. If the result is negative, a zero is shifted into the least significant quotient bit. If the result of the subtraction is positive; the dividend is replaced with the remainder, and a one is shifted into the least significant quotient bit. The dividend is then shifted one bit position to the left, the loop counter is decremented, and the process is repeated until the division is completed. At this point the remainder is tested, and the quotient is rounded up if required.

The dividend exponent is then loaded and incremented to compensate for the extra leading zero in the quotient. If this results in an overflow, an error exit is performend and error code 7 is displayed. The divisor mantissa is subtracted from the dividend mantissa; and depending on the sign of the mantissa, overflow or underflow are tested for. If an error occurs, error code 9 (overflow) or error code 10 (underflow) are displayed. The normalization routine is then called to convert the result to the proper format and add the required mantissa sign. The final quotient remains stored in FPV2.

Floating Point Addition

The floating point addition routine adds the binary floating point numbers stored in FPV1 and FPV2. The two exponents are loaded and compared to determine which is larger. The difference in the exponents is also tested, and if it is large enough, such that the addition of the smaller number would not affect the least significant bits of the larger number, the larger number is stored in the result location (FPV2) and the program is exited. If the numbers are within range of each other, the mantissa of the smaller number is loaded into the two most significant bytes of a four byte workspace. This number is then shifted right by a number of bit positions equal to the difference in exponents. An extra shift is then executed on both mantissas to allow room for an overflow bit. There are two different paths though the program for the above steps, depending on which input variable is larger. Shifting is done using the four byte shift subroutines.

The two mantissas, now normalized to the same exponent, are added using a 32 bit workspace and stored in SHFT1 to SHFT4 for normalization. If the mantissa is negative, it is complemented and the sign flag is set. The normalization routine is then called to produce a valid binary floating point number, which is stored in FPV2.

Floating Point Subtraction

This routine subtracts the minuend stored in FPV2 from the subtrahend stored in FPV1. This is accomplished by taking the two's complement of the minuend mantissa and calling the floating point addition routine to add the two numbers. The difference ends up in FPV2 as usual.

The Four Byte Shift Routines

The four byte shift left routine (FSLST) and the four byte shift right (FSRST) routine will shift a four byte binary number one bit position in the proper direction. The value to be shifted in (a one or a zero) is stored in the carry bit. The two most significant bytes are stored in the HL register pair, while the two least significant bytes are stored on top of the stack, both going into the subroutine and returning from it.

Floating Point Decimal to Binary Conversion

This decimal to binary conversion routine converts the floating point decimal number stored in DECV1 to DECV8 into a floating point binary number. The mantissa sign byte is loaded; if it is set, the sign flag is also set. A decimal digit, starting with the most significant one, is then added to the workspace. The workspace is then multiplied by 10 using binary shifts and addition. This process is repeated until the last decimal digit is encountered. This digit is only added, with no multiplication being performed. In this manner, the mantissa, treated as a decimal integer, is converted into a binary integer. This binary integer is then stored, along with a binary exponent which accounts for the binary point's location to the right of the least significant bit. The normalization routine is then called to convert the mantissa into the proper format. Up to this point, the position of the decimal point has been assumed to be to the right of the least significant decimal digit. Since this is not the case, the binary mantissa is multiplied by 0.001 (decimal) to compensate for this. The decimal exponent sign and magnitude are then loaded. If the decimal exponent is positive, the binary number is multiplied by 10 and the decimal exponent is decremented. This process is repeated until the least significant decimal exponent digit is zero. The binary number is then multiplied by 100 as the most significant decimal exponent digit is decremented; again until the most significant decimal digit is zero. If the decimal exponent is negative, multiplication of the binary number by 0.1 and 0.01 (decimal) is used to account for the decimal exponent. Once the conversion is completed, the resulting binary floating point number is stored in FPV2.

These binary to decimal and decimal to binary conversion routines are generally an order of magnitude slower than the other floating point routines described previously. This delay however does not result in any timing problems, as the conversion routines are only used for microcomputer/operator interfacing.

Floating Point Binary to Decimal Conversion

This routine converts the binary floating point number, stored in FPV2, into a decimal floating point number, storing it in DECV. The binary mantissa is tested; if it is zero, a decimal zero is output and the routine is exited. Otherwise the binary exponent is loaded, incremented to compensate for the mantissa sign bit, and tested. One of two paths is then taken. If the binary exponent is positive, the decimal exponent is incremented and the binary number is multiplied by 0.1 (decimal). This is repeated until the binary exponent is no longer greater than zero. If the binary exponent is negative, the second path is taken. The decimal exponent is decremented and the binary number is multiplied by 10 (decimal), until the binary exponent is non negative. At this point, the binary number and the corresponding decimal exponent obtained before the last cycle through the loop are retrieved, such that a binary exponent is obtained which is as close to zero as possible, but not greater than zero.

Once one of the two above operations has been performed, the binary mantissa is loaded and tested. If it is negative, it is complemented and a sign flag is set in an internal register. The binary exponent is then tested. If the exponent is non zero, it is incremented and the binary mantissa is shifted one bit position to the right. This is repeated until the binary exponent is zero.

The decimal exponent is then stored, the decimal mantissa sign byte is output, and the decimal mantissa is calculated from the shifted binary mantissa as follows. The binary mantissa is now effectively a binary number less than 1.0, with the binary point moved to the left of the most significant bit by earlier incrementing the binary exponent. If this binary number is multiplied by 10 decimal using binary shifting and addition, an overflow between 0 and 9 (decimal) results. This technique is used to shift out five decimal digits, resulting in a decimal number less than 1.0, but equal to the original binary fraction. If the leading decimal digit is a zero, an other digit is shifted out and the stored decimal exponent is decremented. The fifth decimal digit is then used to determine if the fourth digit should be rounded up. If the roundup is performed and results in a decimal overflow, 1.000 is loaded and the stored decimal exponent is incremented. At this point DECV contains the proper decimal mantissa digits.

The decimal exponent is then loaded and decremented to place the decimal point to the right of the most siginficant digit. If the decimal exponent is negative, it is complemented and the sign byte is loaded with the minus sign code. The decimal exponent in binary form is then converted to binary coded decimal form, using the BCD add instruction.

If an overflow error occurs during this process, error exit 14 is performed. Once the BCD form of the decimal exponent is obtained, it is unpacked into two hexadecimal digits and stored in the DECV file; completing the floating point binary to decimal conversion.

C. Calculator Routines

The calculator routines, which enable the CPU system to be used as a four function calculator, occupy memory pages 12 to 15. Keyboard accessible functions are included to enter numbers into the Y register; to add, subtract, multiply and divide; to convert from decimal to hexadecimal; to exchange the X and Y registers; and to store and recall numbers.

The Y Register Load Function

This routine services the LDY key. Upon entry, the floating point workspace is stored using the workspace save routine. This is done in order to avoid problems resulting from the nonreentrant nature of the floating point arithmetic routines. The function flag is cleared, and the decimal data entry routine is called to move the entered decimal data from the display file to DECV, storing it in the proper format. The decimal to binary conversion routine is then called to convert the decimal data stored in DECV into the corresponding floating point binary number. The resulting binary number is then moved to the Y register

(YREG) using the blockmove routine. To confirm the proper entry and conversion of the data, the decimal data display routine is called to display the just entered number. The workspace restore routine is then called to restore the microprocessor's internal registers and the contents of the memory locations affected by the floating point arithmetic routines.

The Decimal Data Entry Routine

This subroutine loads the entered floating point decimal number from the memory file corresponding to the system display, DDAT, and stores it in the DECV file in the proper format. The first character is loaded from the DDAT file. If it is a minus sign flag, it is stored in DECV1; otherwise DECV1 is cleared to indicate a positive mantissa, and the first digit is loaded into DECV2. Up to four mantissa digits are entered into DECV2 to DECV5. The end of the mantissa string is determined in one of the following ways. If four mantissa digits have been entered, the next character is assumed to be either the exponent sign or the first exponent digit. A mantissa with less than four significant digits is indicated if a minus sign flag or a hexadecimal 'E' is encounterd while reading a mantissa digit. If this occurs, the remaining mantissa digits in DECV are set to zero; and the proper sign flag (- for minus sign flag, + for E) is entered into the exponent sign location, DECV6. If a four digit mantissa is used and no exponent sign is encountered, the exponent is assumed to be positive. The character following the fourth mantissa digit, or the exponent sign if one exists, is taken as the single digit exponent magnitude, and is stored in DECV8. DECV7 is cleared, as a two digit decimal exponent will not be used.

The Decimal Data Display Routine

The decimal data display routine takes the floating point number stored in the DECV file, converts it to the proper format, and sends it out to the system's display. The clear flag is set, such that the depression of any data key will clear the display and display the entered data. The exponent magnitude is then entered from DECV7 and 8. If it's magnitude is greater than 9, error code 15 is entered and an error exit is performed; otherwise the exponent is converted to the corresponding seven segment code (using the seven segment conversion routine) and sent to the display. The exponent sign is then displayed (as a - sign or a blank), preceded by an 'E'. The decimal mantissa is displayed, preceded by a '-' sign if required, and the decimal point is entered to the right of the most significant digit.

The Change Sign Function

This routine clears the function flag, then loads the minus sign code and its seven segment equivalent into the BC register pair. The display routine (DSPST) is then called to output the seven segment code to the display, and load the data byte into the corresponding memory file, DDAT.

Decimal to Hexadecimal Conversion

Entered numbers are stored in internal memory locations as floating point binary numbers. The decimal to hexadecimal conversion routine takes the binary number stored in the Y register, unpacks the three bytes into six hexadecimal digits, and outputs them to the display. The hex to seven segment conversion routine is used for the required conversions.

X and Y Register Exchange

This routine clears the function flag, then exchanges the contents of the X (XREG) and Y (YREG) registers. The new variable stored in the X register is converted to decimal using the binary to decimal conversion routine. This number is then displayed using the decimal data display routine. Since the floating point workspace area is affected by this routine, it is saved at the start and restored at the end of the routine, using the subroutine provided for this purpose.

Calculator Multiplication, Division, Addition and Subtraction Functions

These four keyboard service routines (labels CMULS, CDIVS, CADDS and CSUBS) are all identical except for the floating point arithmetic routine called to perform the calculation. The workspace save routine is called to store affected registers and memory locations, then the calculation start routine is called to move the contents of the Y register to FPV1 and load the converted data from the display file to FPV2. The proper arithmetic routine is then called to store the result in the Y register, convert it to decimal number and display it. The workspace restore routine is then called to restore the affected internal registers and memory locations to their initial contents.

The Calculator Function Start Routine

This routine clears the function flag, then calls the decimal data entry routine (DENST) to load the decimal number from the memory display file to the DECV file. The decimal to binary conversion routine is then called to convert the decimal number into a floating point binary number. This new binary number is then stored in the X register, as well as in FPV2, while the content of the Y register is moved to FPV1 using the data block move routine.

The Calculator Function End Routine

This routine moves the result of the calculation to the Y register using the data block move routine. The binary to decimal conversion routine is then called to convert the result to a decimal number, which is output by the decimal data display routine.

The Floating Point Workspace Save Function

Since this routine manipulates the stack pointer, the interrupt system is disabled upon entry and reenabled at the end of the routine. The return address on top of the stack and the HL register pair are exchanged, then the other internal CPU registers are pushed onto the stack. Memory locations affected by the floating point routines (referred to as the floating point workspace) are also pushed onto the stack. A subroutine return is simulated by transferring the return address from the HL register to the program counter.

The Floating Point Workspace Restore Routine

This routine disables the interrupt system, pops the return address from the stack, then restores the workspace and internal register data stored on the stack. The interrupts are then reenabled, the return address is returned to the stack, and a return instruction is executed.

The Data Store Function

This routine services the data store key. The function flag is cleared, and the memory register (0 to 9) is entered from the DDAT file. If the register number is out of range, error code 11 is displayed; otherwise the register number is used to offset a base address. The Y register is then stored in this memory location, the display is cleared using the clear subroutine, and the program is exited.

The Data Recall Function

This routine recalls data stored by the above program. If the requested memory register is out of range, error code 12 is displayed; otherwise the data is copied to FPV2. The binary to decimal conversion and decimal data display routines are then used to display the recalled number. The recalled number is also moved to the DDAT file; thus if the recall is followed by an arithmetic function key, it will appear that the recalled number was actually stored in the X register.

Since the floating point workspace is affected by this routine, the workspace is saved at the beginning of the routine, and restored at completion.

D. Motor Control Software

The following routines, located in memory pages 16 to 23 (decimal) allow the microprocessor system to be used as a DC motor speed controller. The routines are written in a modular manner to simplify modification for other applications.

The Application Module Initialization Routine

This routine is called to initialize the application package, using the monitor load address and run fuctions. The run flag (RUNFL) is cleared to disable the motor control output, and the SCR controlling timer is initialized such that the SCR's are disabled. The sample interval counter and the motor control setpoints and gain constants are loaded with default values, as per figiure 44, using the data block move routine. The auxiliary function stepping stones are then set such that AF0 stops the motor; AF1 starts the motor; and AF2 displays the frequency of the power mains. Before exiting the routine, the display is cleared using the clear routine.

The Motor Start Function

The motor start routine clears the function flag, and tests if a valid entry exists for the maximum allowable steady state current; the RPM setpoint; and if at least one of the two control algorithm gain constants is non zero. If one of the above conditions is not met, a corresponding error message is displayed, otherwise execution of the routine is continued. The time average current (MEM02) and the old integral control value (CVIC) are set to zero. The motor run flag (RUNFL) is then set, and the SCR's are turned full on by entering a 2.0 and calling the control output routine. The control update interrupt (RST 7.5) is enabled, and the sample interval counter location (INTCT) is initialized using the number stored in ICINV.

Motor Control Update

This routine is periodically called by the tachometer interrupt service routine to update the duty cycle of the SCR circuit. Upon entry, the RPM setpoint is tested, and if zero, the motor stop routine is called and the control update program is exited. Otherwise, a series of subroutines is called to read the system RPM (RPM read routine); calculate the new control output (control output routine); and test for armature overcurrent (the current test routine). The sample interval counter is then reinitialized and the internal registers and the floating point workspace are restored.

The Current Test Routine

Instantaneous and I^2t overcurrent are tested for by this routine. The current read routine is called to read the A/D converter. This current is then divided by 5 (since an instantaneous current of five times the steady state limit is allowed) and the result is compared to the steady state current limit stored in MEM08. If the instantaneous limit is exceeded, error code 24 is loaded and an error exit is performed. If the present current is below the instantaneous limit, the present position relative to the I^2t curve is calculated. If this limit is exceeded, error exit 25 is performed; otherwise program control is returned to the control update routine. An error exit in this routine will also call the motor stop routine.

Calculation of the Control Output

This routine loads the RPM setpoint and subtracts from it the present system RPM stored in FPV2. The integral gain constant is tested. If it is zero, the integral control value (CVIC) is also set to zero; otherwise the error is multiplied by the integral gain constant and added to the old integral control variable (CVIC). The result is then tested; if it is less than zero or greater than 2.0, it is clipped at the proper limit. This is then stored in CVIC to be added to the proportional control value, as well as being used as the old integral control value at the next sample interval.

The proportional gain constant is tested, if it is zero, only the integral control value is output. If the proportional gain constant is non zero, the proportional control output is calculated. The proportional control output is then added to the integral control output stored in CVIC. The final result is left in FPV2.

The Control Voltage Output Routine

This routine takes the control value stored in FPV2 and tests it. If the output is zero or negative, the SCR's are turned off and the program is exited; whereas if the control output is greater than or equal to 2.0, the SCR's are turned full on, completing this routine.

A control output between these limits is converted to the required duty cycle count as follows. If the exponent of the control output is equal to 1, the positive binary mantissa is shifted one bit position to the left. A negative exponent is brought to zero by shifting the mantissa to the right and incrementing the exponent until it is zero. This results in a binary mantissa with a value less than 2.0 (decimal) and an exponent equal to zero. Interpolation is then used to obtain from this number, a 16 bit integer corresponding to the number of microseconds which must intervene between the voltage zero and the enabling of the SCR's.

The most significant three bits of the adjusted control output mantissa are used to select one of eight table entries. Each entry consists of a 16 bit base integer and a 16 bit scaling factor. The three most significant bits of the adjusted control output mantissa are masked off and the mantissa is shifted three bit positions to the left, forming a scaling fraction which is multiplied by the scaling factor, using the 16 bit binary multiplication routine. If the scaling fraction is interpreted as a number less than 1.0, and the scaling factor is a 16 bit integer, then the 16 most siginficant bits of the 32 bit integer product can be interpreted as a 16 bit integer. This product integer is then subtracted from the base count to give the proper 16 bit SCR delay count. This count is latched in the timer chip, and will be used until it is updated.

The Motor Stop Routine

This routine clears the function flag, the motor run flag, and disables the SCR controlling counter.

The RPM Conversion Routine

The RPM conversion routine, enters a 16 bit count (RPMCT) corresponding to the number of microseconds elapsed between interrupts originating from the optical tachometer system. The count is tested, and if overrun has occurred, 0.0 RPM is loaded to memory and the routine is exited. If the count is valid, it is complemented, as the counter is a down counter. The count is then incremented by 8 to compensate for the difference between the interrupt to counter initialization delay, and the interrupt to counter latching delay. The result is then normalized to produce a floating point binary number, equal to the number of microseconds required for the motor shaft to rotate through 45 degrees. The system RPM is then calculated by dividing 7.5×10^6 by this count, with the result stored in FPV2 and MEM07.

The SCR Current Conversion Routine

The current conversion routine reads the A/D converter and scales the resulting number to produce a binary floating point number equal to the armature current in amps.

The sample and hold gate is latched. After a delay to allow the optical isolators and the hold capacitor voltage to settle, a start conversion pulse is output. A delay loop gives the returning signal from the end of conversion output on the A/D converter a chance to go to the zero state. The microcomputer then waits for the end of conversion signal to go high, indicating that the conversion is complete. A further delay is added to ensure all optical isolators on the incoming data lines have had a chance to settle. The 8 bit data word is then read, and converted into a valid floating point number using the normalization routine. It is then multiplied by a scaling factor, such that FF (hexadecimal) returned from the A/D converter corresponds to 10.0 (decimal) amps.

Optical Tachometer Interrupt Service Routine

The optical tachometer system generates an interrupt every eighth of a revolution of the motor shaft. These interrupts are serviced by the optical tachometer interrupt routine.

This program keeps track of the number of interrupts using a down counter. When the count reaches 1, a timer/counter is started to measure the time interval to the next interrupt. At the next interrupt, this interval count is used to calculate the system RPM, and update the motor control output voltage. The down counter is then reinitialized for the next sample interval. Figure 75 illustrates the program using a Warnier-Orr diagram [45-46].

The Line Frequency Monitor Routine

This program can be accessed using AF2, once the application module initialization routine has been executed. The display and function flag is cleared, and all maskable interrupts are disabled. The wait subroutine is then called to wait for the rising edge of a zero crossing detector pulse. The timer/counter is then initialized and the following ZCD pulses are counted. On the rising edge of the twentieth pulse, the counter is latched and its contents are read.

```
enter
save internal registers
set SOD, mask RST 7.5
enable other interrupts
load, decrement and store interrupt counter
if counter = 0
    read counter overrun flag
    overrun has occurred
        load overrun code to RPMCT
    ⊕
    overrun has not occurred
        load RPM count to RPMCT
    call control output update subroutine
    clear SOD, reenable RST 7.5
if counter = 1
    restart RPM counter
    reset counter overrun flag
    clear SOD, reenable RST 7.5
restore internal registers
return
```

Figure 75   Optical Tachometer Interrupt Service Routine

The count is then complemented and normalized before being divided into 1.0 x 10^6 to obtain the frequency of the power grid. This value is then displayed using the binary to decimal conversion and decimal data display routines.

VII. Summary and Conclusion

A microcomputer system, useful for digital machine control applications, has been presented. This system consists of several hardware and software modules which can be readily reconfigured to allow the system to be used for various digital machine control applications.

The present system is configured as a DC motor speed control device. The control software consists of a proportional plus integral control program, complemented by an armature current monitoring program. The controller to machine interface consists of a programmable, variable duty cycle, SCR rectifier which provides the motor armature voltage. A current transducer allows the armature current to be monitored. An optical tachometer system allows the actual shaft RPM to be determined, providing the required feedback to the controller.

The operation of the system has been tested by recording the RPM versus time characteristics for various proportional gain, integral gain, and sample interval values. These recordings were taken during motor startup conditions, and during load and RPM setpoint changes. These changes were made under software control, while the motor was running.

The keyboard monitor system permits rapid modification of control parameters, thus making the DC motor speed control system a valuable tool for gaining an insight into the effects of various combinations of control parameters on the system's response. This system readily lends itself to use as a motor speed control demonstration system. It is also easily expanded into a digital synchroncus generator excitation controller for power system stability studies. It is hoped that this device will become a useful addition to the research tools available to the Department of Electrical Engineering.

Bibliography

1. A. L. Blythe, 'The Effects of Static Excitation Characteristics on System Stability', Canadian Electrical Association Meeting, Planning and Operating Section, March 1967.

2. R. J. Fleming, 'Generating Unit Control - An Overview', Canadian Electrical Association Meeting, Toronto, March 1973.

3. J. Lemay, C. Durand, 'Characteristics of Excitation Systems - A Utility Viewpoint', Canadian Electrical Association Meeting, Montreal, March 1969.

4. F. A. Jost, 'State of the Art of Voltage Regulating Systems', Canadian Electrical Association Meeting, Toronto, March 1973.

5. A. Klopfenstein, 'Experience With System Stabilizing Excitation Controls on the Generation of the Southern California Edison Company', IEEE Summer Power Meeting and EHV Conference, Los Angeles, July 1970, Paper No. 70 TP 532-PWR

6. L. Masson, C. R. Desrosiers, 'Field Tests with Stabilizing Signals at Manic 2 - Generating Station', Canadian Electrical Association Meeting, Power System Planning and Operating Section, Montreal, March 1969.

7. D. H. Kelly, M. Gupta, 'Excitation Control for Stability', Canadian Electrical Association Meeting, System Planning and Operating Section, Calgary, October 1969.

8. Yao-Nan Yu, 'Control Strategies for Stabilizing North American Power Systems - Current Practice and Development', Automatic Control, Vol. 2 No. 3, Sept 1974, pp 50-55.

9. A. A. M. El-Ghandakly, *Adaptive Excitation Controller for a Generating Unit*, Ph. D. Thesis, University of Calgary, Dec. 1975.

10. R. M. Maliszewski et al., 'Application of Instrumentation for the Evaluation of Power System Dynamic Performance', IEEE Meeting and International Symposium on High Power Testing, Portland, July 1971, Paper No. 71 TP 579-PWR.

11. A. P. Mane, R. H. Webster, O. J. M. Smith, 'Rotor Velocity Measurment from Power Balance Analog', IEEE

Power Meeting and EHV Conference, Los Angeles, July 1970, Paper No. 70 TP 678-PWR.

12. R. N. Sudan, V. N. Manohar, B. Adkins, 'The Measurment of Transient Torques and Load Angle in Model Synchronous Machines', IEE Paper No. 3151 S, Feb. 1960.

13. J. P. Bartlett, M. J. Gibband, J. L. Woodward, 'Performance of a 5 KVA Synchronous Generator with an Optimal Excitation Regulator', Proc. of IEE, Vol. 120, No. 10, Oct. 1973, pp 1250-1256.

14. W. S. Leung, Y. S. Cheung, Y. C. Yeung, 'Computerized Linear Control of A. C. Generator Power Output', Proc. of IEE, Vol. 121, No. 10, pp 1150-1154, Oct. 1974.

15. D. B. Mehta, B. Adkins, 'Transient Torque and Load Angle of a Synchronous Generator Following Several Types of System Disturbances', IEE Paper No. 3160 S, Feb. 1960.

16. T. L. Dillman et al., 'A High Initial Response Brushless Excitation System', IEEE Power Meeting, New York, Jan. 1971, Paper No. 71 TP 29-PWR.

17. E. J. Davison, N. S. Rau, 'The Optimal Output Feedback Control of a Synchronous Machine', IEEE Power Meeting, New York, Jan. 1971, Paper No. 71 TP 102-PWR.

18. H. H. Hwang, P. C. Tripathi, 'Stability Analysis and Synthesis of Generator Excitation Control', IEEE Summer Meeting and International Symposiym on High Power Testing, Portland, July 1971, Paper No. 71 CP 621-PWR. (A revised version of this paper appeared in Automatic Control, Vol. 2, No. 3, Sept. 1974.)

19. F. L. Stromotich, R. J. Fleming, 'Generator Damping Enhancement Using Suboptimal Control Methods', IEEE PES Meeting, San Francisco, July 1972, Paper No. C 72 472-9.

20. C. E. Grund, 'Dynamic Stability Enhancement With Power System Stabilizers', IEEE PES Meeting, New York, Jan. 1974, Paper No. C 74 123-6.

21. L. J. Jacovides, B. Adkins, 'Effect of Excitation Regulation on Synchronous Machine Stability', Proc. IEE, Vol. 113, No. 6, June 1966, pp 1021-1034.

22. J. L. Dineley, 'The Present Feasability of on Line Computer Control of Synchronous Generator Stability', IEEE Power Meeting, New York, Jan 1970, Paper No. 70 CP 150-PWR.

23. A. H. M. A. Rahim, A Quasi-Optimal Excitation Control for Power System Stability, Ph.D. Thesis, Department of Electrical Engineering, University of Alberta, Spring 1972.

24. M. E. Newton, B. W. Hogg, 'Optimal Control of a Micro Alternator System', IEEE Trans. on Power Apparatus and Systems, Vol. 95, No. 6, Nov./Dec. 1976.

25. A. H. M. A. Rahim, D. H. Kelly, 'Optimal Excitation Control for Power System Stability', IEEE Trans. on Automatic Control, Vol. 16, No. 4, Aug. 1971, pp 379-381.

26. A. H. M. A. Rahim, D. H. Kelly, 'Evaluation of a Quasi Optimal State Feedback Excitation Strategy for Power System Stability', IEEE Trans. on Automatic Control, Vol. 17, No. 4, Aug. 1972, pp 515-518.

27. D. H. Kelly, A. H. M. A. Rahim, 'A Quasi-Optimal State Feedback Excitation Control for a Two Machine System', IEEE PES Meeting, San Francisco, July 1972, Paper No. C 72 488-5.

28. D. H. Kelly, A. H. M. A. Rahim, 'Closed Loop Optimal Excitation Control for Power System Stability', IEEE Trans. on Power Apparatus and Systems, Vol. 90, No. 5, Sept./Oct. 1971, pp 2135-2141.

29. A. H. M. A. Rahim, D. H. Kelly, V. Gourishankar, 'Optimal, Sub-Optimal and Practical Excitation Control Schemes for Stability', IEEE PES Meeting, New York, January 1973, Paper No. C 73 081-7.

30. A. E. Fitzgerald et al., Electric Machinery, McGraw Hill, New York, 1971.

31. A. K. Lin, W. W. Koepsel, 'A Microprocessor Speed Control System', IECI '77 Proc., Industrial Electronics and Control Instrumentation, March 21-23, 1977, Philadelphia, Penn.

32. A. S. A. Farag, O. P. Malik, G. S. Hope, 'Studies on an SCR Controlled Variable Speed DC Shunt Motor', IEEE Trans. on Power Apparatus and Systems, Vol. 93, Pt. 1, 1974, pp 785-792.

33. A. M. Roy, P. K. Sen, H. Banerjee, 'Speed Control of DC Shunt Motors by Solid State Devices', Indian Journal of Technology, Vol. 13, April 1975, pp 160-164.

34. T. J. Maloney, F. L. Alvarado, 'A Digital Method for DC Motor Speed Control', IEEE Trans. on Industrial Electronics and Control Instrumentation, Vol. 23, No. 1, Feb. 1976, pp 44-46.

35. L. A. Philips, W. L. Green, 'Development and Performance of a Microcomputer for Closed Loop Digital Control', Proc. of 1976 IEEE Southeastcon Region 3 Conference on Engineering in a Changing Economy, pp 253-255.

36. R. C. Blach, T. W. Reddoch, 'Solid State DC Motor Control', Proc. of 1976 IEEE Southeastcon Region 3 Conference on Engineering in a Changing Economy, pp 168-170.

37. *Semiconductor Power Circuits Handbook*, Motorola, Semiconductor Products Division, 1968.

38. S. A. Farag, 'State-Space Approach to the Analysis of DC Machines Controlled by SCRs', IEEE 1976 Control of Power Systems Conference and Exposition, Conference Record, Oklahoma City, pp 157-163.

39. H. Birck, 'Digitale Regelung durch Mikroprozessoren', Elektronik, Sonderheft 1, Mikroprozessoren; Hardware, pp 98-100.

40. K. M. Nohr, *A Microprocessor Controller System*, M. Sc. Thesis, University of Alberta, Fall 1978.

41. R. J. Hall, M. Dean, 'LEDs Work Well with an AC Line in New Zealand', Electronics, Nov. 25, 1976, p 134.

42. D. Johnson, J. Hilburn, *Rapid Practical Design of Active Filters* , Wiley, New York.

43. T. G. Lewis, J. W. Doerr, *Minicomputers: Structures and Programming*, Hayden, New Jersey, 1976.

44. R. Findley, *Scelbi '8080' Software Gourmet Guide and Cook Book*, Scelbi Computer Consulting Inc., 1976.

45. D. A. Higgins, 'Structured Programming with Warnier-Orr Diagrams: Part 1', Byte, Dec. 1977, pp 104-110.

46. D. A. Higgins, 'Structured Programming with Warnier-Orr Diagrams: Part 2', Byte, Jan. 1978, pp 122-129.

Appendix A System Software Listing

This section consists of an assembled listing of all system software, formated in the following manner. The first four hexadecimal digits constitute the memory address of the first byte of the following opcode or data field. The address field is followed by two to eight opcode or data digits (representing one to four bytes). If the line contains a label, it occupies the next five character locations, followed by a colon. The opcode menomic starts in column 19, followed by the operand if one is required. Column 32 contains the first character of the comment field.

A. Address and Label Definitions

In the following lines, the address labels associated with data storage locations and I/O ports are defined.

Monitor Data File

```
2000  SSAF0   JMP TO AUX FTN 0
2003  SSAF1   JMP TO AUX FTN 1
2006  SSAF2   JMP TO AUX FTN 2
2009  SSAF3   JMP TO AUX FTN 3
200C  DATAF   DATA ENTER FLAG
200D  DISCT   DISPLAY ADDRESS OFFSET
200E  FTNFL   FUNCTION FLAG
200F  ADDRL   MONITOR ADDRESS BUFFER
2010  ADDRH
2011  DDAT0   DISPLAY DATA FILE
2012  DDAT1
2013  DDAT2
2014  DDAT3
2015  DDAT4
2016  DDAT5
```

2017	DDAT6	
2018	DDAT7	
2019	IOSB0	I/O READ WRITE ROUTINE
201A	IOSB1	SCRATCHPAD
201B	IOSB2	
201C	FPV11	FLOATING POINT PACK
201D	FPV12	BINARY SCRATCHPAD
201E	FPV13	
201F	FPV21	
2020	FPV22	
2021	FPV23	
2022	SIGNF	NORMALIZATION SIGN FLAG
2023	SHFT1	NORMALIZATION ROUTINE DATA
2024	SHFT2	
2025	SHFT3	
2026	SHFT4	
2027	SHFT5	
2028	FPLCT	FLOATING POINT PACK LOOP COUNTER
2029	DECV1	DECIMAL DATA FILE MANTISSA SIGN
202A	DECV2	MANTISSA
202B	DECV3	
202C	DECV4	
202D	DECV5	
202E	DECV6	EXPONENT SIGN
202F	DECV7	EXPONENT
2030	DECV8	
2031	XREG1	X REGISTER
2032	XREG2	
2033	XREG3	
2034	YREG1	Y REGISTER
2035	YREG2	
2036	YREG3	
2037	CLRFL	CLEAR FLAG
2038	MEM00	FIRST STORAGE LOCATION
203B	MEM01	
203E	MEM02	
2041	MEM03	
2044	MEM04	
2047	MEM05	
204A	MEM06	
204D	MEM07	
2050	MEM08	
2053	MEM09	
2056	FPDTS	TEMP STORAGE FOR F.P. DATA STORE/RECALL ROUTINES

Application Program Data File

2058	RPMCT	LAST RPM COUNT
205A	CVIC1	OLD INTEGRAL CONTROL
205D	TIMCT	CONTROL UPDATE TIMER COUNT
205F	CUTR1	LAST RPM ERROR
2062	RUNFL	MOTOR RUN FLAG
2064	INTCT	CONTROL INTERRUPT COUNTER
2065	ICINV	INITIAL VALUE OF ABOVE
2066	DISPF	MEMORY REGISTER DISPLAY FLAG

I/O Port Addresses

0004	D8279	DISP DATA REG
0005	C8279	DISP CONTROL REG
0006	C8214	INTERRUPT DECODER
00F0	ADC01	A/D CONVERTER DATA
00F1	FFDAT	FLIP FLOP DATA
00F3	TRS12	TIMER OVERRUN RESET
00F4	LATSH	LATCH SAMPLE/HOLD
00F7	TRS11	CONTROL UPDATE TIMER INTERRUPT RESET
00FC	TIM10	TIMER CHIP 1, COUNTER 0
00FD	TIM11	TIMER CHIP 1, COUNTER 1
00FE	TIM12	TIMER CHIP 1, COUNTER 2
00FF	TIM1C	TIMER CHIP 1, CONTROL REGISTER

B. The Monitor System

Interrupt Vectors

```
0000 C38300        JMP RSTST     RESET VECTOR AND
                                 INTERRUPT 0
0003 000000        DB 0H,0H,0H
0006 0000          DB 0H,0H
0008 C35500        JMP INEST     INTERRUPT 1
000B 000000        DB 0H,0H,0H
000E 0000          DB 0H,0H
0010 C35500        JMP INEST     INTERRUPT 2
0013 000000        DB 0H,0H,0H
0016 0000          DB 0H,0H
0018 C35500        JMP INEST     INTERRUPT 3
001B 000000        DB 0H,0H,0H
001E 0000          DB 0H,0H
0020 C35500        JMP INEST     INTERRUPT 4
0023 00            DB 00H
0024 C35500        JMP INEST     TRAP
0027 00            DB 00H
0028 C35500        JMP INEST     INTERRUPT 5
002B 00            DB 00H
002C C3CB00        JMP KBDST     RST 5.5 - KEYBOARD
002F 00            DB 00H
0030 C35500        JMP INEST     INTERRUPT 6
0033 00            DB 00H
0034 C35500        JMP INEST     RST 6.5
0037 00            DB 00H
0038 C35500        JMP INEST     INTERRUPT 7
003B 00            DB 00H
003C C30314        JMP RPMIF     RST 7.5 - RPM PULSER
003F 00            DB 00H
```

Forwarding Addresses

```
0040 C35D00 ERRFA:JMP ERRST    ERROR ROUTINE
0043 C38300 RSTFA:JMP RSTST    RESET
0046 C3CB00 KBDFA:JMP KBDST    KEYBOARD MONITOR
0049 C3B601 CLRFA:JMP CLRST    DISPLAY CLEAR
004C C3A803 CVSFA:JMP CVSST    HEX TO SEVEN SEG
004F C3D103 BKMFA:JMP BKMST    BLOCK MOVE
0052 C34202 DSPFA:JMP DSPST    DISPLAY WRITE
```

Interrupt Error Routine

```
                               UNDEFINED INTERRUPTS RESULT
                               IN JUMP TO INTER ROUTINE
0055 F5      INEST:PUSH PSW    SAVE ACCUMULATOR
0056 E5            PUSH H      AND HL PAIR
0057 210060        LXI H,6000H LOAD ERROR CODE, 1
005A C35D00        JMP ERRST   JUMP TO ERROR DISPLAY ROUTINE
```

Error Exit Routine

```
                               ERROR DISPLAY ROUTINE
                               DISPLAY ERROR CODE AND
                               GO INTO HALT STATE
                               HL - ERROR NUMBER, DECODED
005D 3E90    ERRST:MVI A,90H   PREPARE 8279 TO WRITE
005F D305          OUT C8279   WITH AUTO INCREMENT ON
0061 7D            MOV A,L     OUTPUT ERROR CODE
0062 D304          OUT D8279
0064 7C            MOV A,H
0065 D304          OUT D8279
0067 AF            XRA A       WRITE SPACE
0068 D304          OUT D8279
006A 3E05          MVI A,05H   WRITE R
006C D304          OUT D8279
006E 3E97          MVI A,97H   WRITE E
0070 D304          OUT D8279
0072 AF            XRA A       CLEAR A
0073 D304          OUT D8279   CLEAR NEXT 3 DIGITS
0075 D304          OUT D8279
0077 D304          OUT D8279
0079 20            RIM         READ INT MASK
007A E606          ANI 06H     MASK FOR RST 6.5, 7.5
```

```
                                         CLEAR SOD
007C C648           ADI 48H              ENABLE RST 5.5
007E 30             SIM                  WRITE INT MASK
007F FB             EI                   ENABLE INTERRUPTS
0080 E1             POP H                RESTORE REGISTERS
0081 F1             POP PSW
0082 76             HLT                  WHOA
```

System Reset Routine

```
                                         INITIALIZE SYSTEM
0083 31FF23 RSTST:LXI SP,23FFH           LOAD STACK POINTER
0086 3E00           MVI A,00             SET KEYBOARD TO 2
0088 D305           OUT C8279            KEY LOCKOUT
                                         DISP TO 8 DIG LEFT ENTRY
008A 3E24           MVI A,24H            SET DIVIDER TO 04D
008C D305           OUT C8279
008E 3ED2           MVI A,0D2H           CLEAR FIFO, DISP RAM
0090 D305           OUT C8279
0092 DB05    RST01:IN C8279              READ 8279 CONDITION CODE
0094 A7             ANA A                TEST
0095 FA9200         JM RST01             LOOP IF DISP RAM UNAVAILABLE
0098 CDB601         CALL CLRST           CLEAR DISP, FLAGS, DISP FILE

                                         DISPLAY READY
                                         -------------
009B 3E87           MVI A,87H            SELECT LEFT DIGIT
009D D305           OUT C8279
009F 3E05           MVI A,05H            OUTPUT R
00A1 D304           OUT D8279
00A3 3EFF           MVI A,0FFH           SET CLEAR FLAG
00A5 323720         STA CLRFL
00A8 3E5E           MVI A,5EH            ENABLE RST SYSTEM
                                         ENABLE RST 5.5
                                         DISABLE REST
                                         CLEAR SOD
00AA 30             SIM                  SET INTERRUPT MASK

                                         INITIALIZE STEPPING STONES
                                         --------------------------
00AB 110020         LXI D,SSAF0          LOAD START ADDRESS
00AE 2E04           MVI L,04H            LOAD NO OF STEPPING STONES
00B0 01C000 RST02:LXI B,RSTTB            LOAD START OF DATA TABLE
00B3 2603           MVI H,03H            LOAD NO OF DATA BYTES
00B5 CDD103         CALL BKMST           MOVE TO RAM
00B8 2D             DCR L                DECREMENT COUNTER
00B9 C2B000         JNZ RST02            LOAD DATA TO NEXT
                                         STEPPING STONE
00BC FB             EI                   ENABLE INTERRUPT SYSTEM
```

```
00BD C3C703       JMP LOPST    GO TO IDLE LOOP

                               DATA TABLE
                               ----------
00C0 C3C300 RSTTB:JMP RST03    RETURN TO ERROR ROUTINE

                               STEPPING STONE ERROR
                               --------------------
00C3 F5     RST03:PUSH PSW
00C4 E5           PUSH H
00C5 21F460       LXI H,60F4H  LOAD ERROR CODE 13
00C8 C35D00       JMP ERRST
```

<u>The Keyboard Interrupt Service Routine</u>

```
                               KEYBOARD READ ROUTINE
                               ---------------------
00CB F5     KBDST:PUSH PSW     SAVE INTERNAL REGISTERS
00CC C5           PUSH B       ON STACK
00CD D5           PUSH D
00CE E5           PUSH H
00CF 20           RIM          READ INT MASK
00D0 E607         ANI 07H      MASK FOR RST MASKS
00D2 C6C9         ADI 0C9H     DISABLE RST 5.5, SET SOD
00D4 30           SIM          WRITE MASK
00D5 FB           EI           ENABLE INTERRUPTS
00D6 3E40         MVI A,40H    LOAD READ COMMAND
00D8 D305         OUT C8279    OUTPUT TO 8279
00DA DB04         IN  D8279    READ FIFO
00DC 4F           MOV C,A      SAVE
00DD 3A0E20       LDA FTNFL    LOAD FTN FLAG
00E0 81           ADD C        ADD KEY CODE
00E1 211901       LXI H,KBDTB  LOAD ADD OF KEYBOARD TABLE
00E4 062E         MVI B, 2EH   LOAD MAX NO OF TABLE
                               ENTRIES, 46 D
00E6 BE     KBD01:CMP M        COMPARE CODE TO TABLE
00E7 CAF900       JZ KBD02     CORRECT ENTRY FOUND
00EA 23           INX H        INCREMENT HL TO NEXT CODE
00EB 23           INX H
00EC 23           INX H
00ED 05           DCR B        DECREMENT COUNTER
00EE C2E600       JNZ KBD01    IF NOT ZERO, CHECK NEXT
                               TABLE ENTRY
                               KEYBOARD ERROR
00F1 F5           PUSH PSW     SAVE REGISTERS
00F2 E5           PUSH H
00F3 2100F4       LXI H,0F400H LOAD ERROR CODE, 3
00F6 C35D00       JMP ERRST    JUMP TO ERROR DISP ROUTINE
00F9 23     KBD02:INX H        INCREMENT HL AND
```

```
00FA 5E           MOV E,M       LOAD ADDRESS
                                NOTE THAT ASSEMBLER STORES
                                LS BYTE FIRST
00FB 23           INX H
00FC 56           MOV D,M
00FD 01A301       LXI B,KBDRT   STORE RETURN ADD
0100 C5           PUSH B        IN STACK
0101 D5           PUSH D        LOAD JUMP ADDRESS
0102 F3           DI            DISABLE INT TO PREVENT
                                ERROR DUE TO INT DURING
                                STACK MANIPULATION
0103 33           INX SP        POINT TO REGISTER DATA
0104 33           INX SP
0105 33           INX SP
0106 33           INX SP
0107 E1           POP H         RESTORE REGISTERS FOR
0108 D1           POP D         SERVICE ROUTINES
0109 C1           POP B
010A F1           POP PSW
010B 3B           DCX SP        RESTORE STACK POINTER
010C 3B           DCX SP
010D 3B           DCX SP
010E 3B           DCX SP
010F 3B           DCX SP
0110 3B           DCX SP
0111 3B           DCX SP
0112 3B           DCX SP
0113 3B           DCX SP
0114 3B           DCX SP
0115 3B           DCX SP
0116 3B           DCX SP
0117 FB           EI            REENABLE INTERRUPTS
0118 C9           RET           JUMP TO STORED ADDRESS
                                NOTE, THIS EXIT SIMULATES
                                A SUBROTINE CALL

                                KEYBOARD LOOKUP TABLE
                                ---------------------
0119 00     KBDTB:DB 00H        KEY CODE
011A 0C02         DW BUT07      ADD OF SERVICE ROUTINE
011C 01           DB 01H
011D 1202         DW BUT08
011F 02           DB 02H
0120 1802         DW BUT09
0122 03           DB 03H
0123 3602         DW BUT0E
0125 04           DB 04H
0126 3C02         DW BUT0F
0128 05           DB 05H
0129 AE01         DW KBD03
012B 08           DB 08H
012C FA01         DW BUT04
012E 09           DB 09H
012F 0002         DW BUT05
```

```
0131 0A               DB 0AH
0132 0602             DW BUT06
0134 0B               DB 0BH
0135 2A02             DW BUT0C
0137 0C               DB 0CH
0138 3002             DW BUT0D
013A 0D               DB 0DH
013B 0803             DW RUNST
013D 10               DB 10H
013E E801             DW BUT01
0140 11               DB 11H
0141 EE01             DW BUT02
0143 12               DB 12H
0144 F401             DW BUT03
0146 13               DB 13H
0147 1E02             DW BUT0A
0149 14               DB 14H
014A 2402             DW BUT0B
014C 15               DB 15H
014D 7202             DW LDAST
014F 18               DB 18H
0150 1803             DW SFFST
0152 19               DB 19H
0153 E201             DW BUT00
0155 1A               DB 1AH
0156 9502             DW MWRST
0158 1B               DB 1BH
0159 0003             DW MRDST
015B 1C               DB 1CH
015C B601             DW CLRST
015E 80               DB 80H
015F 000C             DW CENTF
0161 81               DB 81H
0162 AE01             DW KBD03
0164 82               DB 82H
0165 030C             DW CCGSF
0167 83               DB 83H
0168 090C             DW CSXYF
016A 84               DB 84H
016B 180C             DW CSTRF
016D 85               DB 85H
016E 1B0C             DW CRCLF
0170 88               DB 88H
0171 120C             DW CADDF
0173 89               DB 89H
0174 150C             DW CSUBF
0176 8A               DB 8AH
0177 0C0C             DW CMULF
0179 8B               DB 8BH
017A 0F0C             DW CDIVF
017C 8C               DB 8CH
017D 060C             DW CHEXF
017F 8D               DB 8DH
0180 AE01             DW KBD03
```

```
0182 90                 DB 90H
0183 0320               DW SSAF1
0185 91                 DB 91H
0186 0620               DW SSAF2
0188 92                 DB 92H
0189 0920               DW SSAF3
018B 93                 DB 93H
018C AE01               DW KBD03
018E 94                 DB 94H
018F AE01               DW KBD03
0191 95                 DB 95H
0192 AE01               DW KBD03
0194 98                 DB 98H
0195 AE01               DW KBD03
0197 99                 DB 99H
0198 0020               DW SSAF0
019A 9A                 DB 9AH
019B 7C03               DW IWRST
019D 9B                 DB 9BH
019E 1E03               DW IRDST
01A0 9C                 DB 9CH
01A1 B601               DW CLRST

                                     RETURN ROUTINE
                                     --------------
01A3 20        KBDRT:RIM             READ INT MASK
01A4 E606               ANI 06H      MASK FOR RST MASKS
01A6 C648               ADI 48H      ENABLE RST 5.5, CLEAR SOD
01A8 30                 SIM          WRITE MASK
01A9 E1                 POP H        RETURN FROM KEYBOARD SERVICE
                                     ROUTINES
01AA D1                 POP D        RESTORE REGISTERS, EVEN
01AB C1                 POP B        IF MESSED UP BY
01AC F1                 POP PSW      SERVICE ROUTINE
01AD C9                 RET          RETURN

                                     ERROR ROUTINE FOR
                                     UNDEFINED KEY FTN
01AE F5        KBD03:PUSH PSW        SAVE REGISTERS
01AF E5                 PUSH H
01B0 2100B5             LXI H,0B500H LOAD CODE, 2
01B3 C35D00             JMP ERRST    JMP TO ERROR DISP ROUTINE
```

The Display Clear Routine

```
01B6 F5      CLRST:PUSH PSW      SAVE INT REGISTERS
01B7 C5            PUSH B
01B8 D5            PUSH D
01B9 E5            PUSH H
01BA 3E07          MVI A,07H     SET DIGIT COUNT TO 7
01BC 320D20        STA DISCT
01BF 3E90          MVI A,90H     AUTO INC DISP RAM WRITE
01C1 D305          OUT C8279
01C3 AF            XRA A         CLEAR A
01C4 320E20        STA FTNFL     CLEAR FLAGS
01C7 320C20        STA DATAF
01CA 323720        STA CLRFL
01CD 326620        STA DISPF
01D0 0608          MVI B,08H     LOAD COUNTER
01D2 211120        LXI H,DDAT0   LOAD POINTER
01D5 77      CLR01:MOV M,A       CLEAR DISPLAY FILE
01D6 D304          OUT D8279     CLEAR DISP RAM
01D8 23            INX H         INCREMENT POINTER
01D9 05            DCR B         DECREMENT COUNTER
01DA C2D501        JNZ CLR01     REPEAT UNTIL DONE
01DD E1            POP H         RESTORE INT REGISTERS
01DE D1            POP D
01DF C1            POP B
01E0 F1            POP PSW
01E1 C9            RET
```

Data Entry Routines

```
01E2 01F300 BUT00:LXI B,00F3H    LOAD HEX NUMBER AND
                                 SEVEN SEGMENT CODE
01E5 C34202       JMP DSPST      JUMP TO DISPLAY ROUTINE
01E8 016001 BUT01:LXI B,0160H
01EB C34202       JMP DSPST
01EE 01B502 BUT02:LXI B,02B5H
01F1 C34202       JMP DSPST
01F4 01F403 BUT03:LXI B,03F4H
01F7 C34202       JMP DSPST
01FA 016604 BUT04:LXI B,0466H
01FD C34202       JMP DSPST
0200 01D605 BUT05:LXI B,05D6H
0203 C34202       JMP DSPST
0206 01D706 BUT06:LXI B,06D7H
0209 C34202       JMP DSPST
020C 017007 BUT07:LXI B,0770H
020F C34202       JMP DSPST
0212 01F708 BUT08:LXI B,08F7H
```

```
0215 C34202        JMP DSPST
0218 01F609 BUT09:LXI B,09F6H
021B C34202        JMP DSPST
021E 01770A BUT0A:LXI B,0A77H
0221 C34202        JMP DSPST
0224 01C70B BUT0B:LXI B,0BC7H
0227 C34202        JMP DSPST
022A 01930C BUT0C:LXI B,0C93H
022D C34202        JMP DSPST
0230 01E50D BUT0D:LXI B,0DE5H
0233 C34202        JMP DSPST
0236 01970E BUT0E:LXI B,0E97H
0239 C34202        JMP DSPST
023C 01170F BUT0F:LXI B,0F17H
023F C34202        JMP DSPST
                                 ROUTINE TO ENTER DATA INTO
                                 DISPLAY AND RAM
                                 B - HEX DIGIT
                                 C - 7 SEG CODE
0242 3A3720 DSPST:LDA CLRFL      LOAD CLEAR FLAG
0245 A7            ANA A         TEST
0246 C4B601        CNZ CLRST     CALL CLEAR ROUTINE IF SET
0249 3A0D20        LDA DISCT     LOAD DISP ADD OFFSET
024C 5F            MOV E,A
024D C680          ADI 80H
024F D305          OUT C8279     SELECT RAM LOCATION
0251 79            MOV A,C
0252 D304          OUT D8279     WRITE DIGIT
0254 1600          MVI D,00H     CLEAR D
0256 211120        LXI H,DDAT0   LOAD ADD OF DDAT0
0259 19            DAD D         ADD BC AND HL PAIRS
025A 70            MOV M,B       LOAD STORED DIGIT
                                 TO SCRATCHPAD
025B 1D            DCR E         DECREMENT DISPLAY COUNT
025C F26D02        JP DSP02      SKIP IF POSITIVE
025F 3A0C20        LDA DATAF     LOAD DATA/ADD ENTER FLAG
0262 BA            CMP D         CHECK IF ZERO
0263 CA6B02        JZ DSP01      SKIP IF DATA FLAG NOT SET
0266 1E01          MVI E,01H     SET DISP COUNT TO 1
0268 C36D02        JMP DSP02
026B 1E07   DSP01:MVI E,07H      SET DISP COUNT TO 7
026D 7B     DSP02:MOV A,E
026E 320D20        STA DISCT     STORE DECREMENTED DISP COUNT
0271 C9            RET
```

The Load Address Routine

```
0272 211820 LDAST:LXI H,DDAT7  LOAD ADD OF FIRST DATA ENTRY
0275 7E           MOV A,M      LOAD MS NIBBLE
0276 07           RLC
0277 07           RLC
0278 07           RLC
0279 07           RLC
027A 2B           DCX H
027B 86           ADD M        ADD 2ND NIBBLE
027C 57           MOV D,A
027D 2B           DCX H
027E 7E           MOV A,M      LOAD 3RD NIBBLE
027F 07           RLC
0280 07           RLC
0281 07           RLC
0282 07           RLC
0283 2B           DCX H
0284 86           ADD M        ADD LS NIBBLE
0285 5F           MOV E,A
0286 EB           XCHG         MOVE DE TO HL
0287 CDB402       CALL MDSST   STORE ADD, DISP DATA
028A 3EFF         MVI A,0FFH
028C 320C20       STA DATAF    SET DATA FLAG
028F 3E01         MVI A,01H
0291 320D20       STA DISCT    SET DIGIT COUNT TO 1
0294 C9           RET
```

The Memory Write Function

```
0295 211220 MWRST:LXI H,DDAT1  LOAD ADDRESS OF MS
                               DATA NIBBLE
0298 7E           MOV A,M
0299 07           RLC
029A 07           RLC
029B 07           RLC
029C 07           RLC
029D 2B           DCX H        DECREMENT HL
029E 86           ADD M        ADD LS NIBBLE
029F 2A0F20       LHLD ADDRL   LOAD DATA ADDRESS
02A2 77           MOV M,A      LOAD NEW DATA TO MEMORY
02A3 BE           CMP M
02A4 CAAF02       JZ MWR01     SKIP IF DATA WRITTEN OK
02A7 F5           PUSH PSW     ERROR EXIT
02A8 E5           PUSH H
02A9 21F3B5       LXI H,0B5F3H ERROR CODE 20
02AC C35D00       JMP ERRST
02AF 23     MWR01:INX H        INCREMENT DATA ADDRESS
02B0 CDB402       CALL MDSST   RD AND DISP DATA
02B3 C9           RET
```

Reading and Displaying a Memory Location

```
02B4 46     MDSST:MOV B,M         READ MEMORY LOCN
                                  B - DATA
                                  HL - ADDRESS
02B5 220F20       SHLD ADDRL      STORE MEMORY LOCN ADDR
02B8 3E90         MVI A,90H       PREPARE DISP RAM FOR NEW DATA
02BA D305         OUT C8279       WITH AUTO INCR ON
02BC 78           MOV A,B
02BD E60F         ANI 0FH         MASK
02BF CDA803       CALL CVSST
02C2 D304         OUT D8279       WRITE TO DISP
02C4 78           MOV A,B
02C5 0F           RRC
02C6 0F           RRC
02C7 0F           RRC
02C8 0F           RRC
02C9 E60F         ANI 0FH
02CB CDA803       CALL CVSST
02CE D304         OUT D8279
02D0 7D           MOV A,L
02D1 E60F         ANI 0FH
02D3 CDA803       CALL CVSST
02D6 C608         ADI 08H         ADD DOT
02D8 D304         OUT D8279
02DA 7D           MOV A,L
02DB 0F           RRC
02DC 0F           RRC
02DD 0F           RRC
02DE 0F           RRC
02DF E60F         ANI 0FH
02E1 CDA803       CALL CVSST
02E4 D304         OUT D8279
02E6 7C           MOV A,H
02E7 E60F         ANI 0FH
02E9 CDA803       CALL CVSST
02EC D304         OUT D8279
02EE 7C           MOV A,H
02EF 0F           RRC
02F0 0F           RRC
02F1 0F           RRC
02F2 0F           RRC
02F3 E60F         ANI 0FH
02F5 CDA803       CALL CVSST
02F8 D304         OUT D8279
02FA AF           XRA A           CLEAR A
```

```
02FB D304          OUT D8279    CLEAR 2 LEFT DIGITS
02FD D304          OUT D8279
02FF C9            RET
```

The Memory Read Function

```
0300 2A0F20 MRDST:LHLD ADDRL    LOAD DATA ADDRESS TO HL
0303 23           INX H         INCREMENT ADDRESS
0304 CDB402       CALL MDSST    LOAD AND DISP DATA
0307 C9           RET
```

The Run Function

```
0308 2600   RUNST:MVI H,00H     LOAD STACK POINTER OFFSET
030A 2E0A         MVI L,0AH
030C 39           DAD SP        ADD SP TO HL
030D F9           SPHL          LOAD RESULT TO STACK POINTER
030E 2A0F20       LHLD ADDRL    LOAD ADDRESS
0311 20           RIM           READ INT MASK
0312 E606         ANI 06H       MASK FOR RST MASKS
0314 C648         ADI 48H       ENABLE RST 5.5, CLEAR SOD
0316 30           SIM           WRITE MASK
0317 E9           PCHL          LOAD HL TO PC, JUMP SHIP
```

Setting the Function Flag

```
0318 3E80   SFFST:MVI A,80H
031A 320E20       STA FTNFL     SET FTN FLAG TO 80
031D C9           RET
```

The I/O Port Read Function

```
031E AF       IRDST:XRA A         CLEAR A
031F 320E20         STA FTNFL     CLEAR FTN FLAG
0322 211820         LXI H,DDAT7   LOAD ADDRESS OF DATA
0325 7E             MOV A,M       ENTER MS NIBBLE
0326 07             RLC
0327 07             RLC
0328 07             RLC
0329 07             RLC
032A 2B             DCX H
032B 86             ADD M         ADD LS NIBBLE
032C 47             MOV B,A       STORE
032D 211920         LXI H,IOSB0   LOAD ADD OF I/O
                                  ROUTINE SPACE
0330 3EDB           MVI A,0DBH    OPCODE FOR I/O READ
0332 77             MOV M,A       STORE
0333 23             INX H
0334 70             MOV M,B       STORE PORT NO
0335 3EC9           MVI A,0C9H    OPCODE FOR RET
0337 23             INX H
0338 77             MOV M,A
0339 CD1920         CALL IOSB0    CALL JUST WRITTEN ROUTINE
033C 4F             MOV C,A       STORE PORT CONTENTS
033D CD4103         CALL IDSST
0340 C9             RET
```

I/O Port Data Display Routine

```
                                  B - PORT NO
                                  C - CONTENTS OF PORT
0341 3E90     IDSST:MVI A,90H     SET UP 8279 FOR DATA WRITE
                                  WITH AUTO INCREMENT
0343 D305           OUT C8279
0345 AF             XRA A         CLEAR A
0346 D304           OUT D8279     WRITE TO LOCN 0
0348 79             MOV A,C       LOAD CONTENTS OF PORT
0349 E60F           ANI 0FH       MASK
034B CDA803         CALL CVSST
034E D304           OUT D8279
0350 79             MOV A,C
0351 0F             RRC
0352 0F             RRC
0353 0F             RRC
0354 0F             RRC
0355 E60F           ANI 0FH
0357 CDA803         CALL CVSST
035A D304           OUT D8279
```

```
035C 78               MOV A,B      LOAD PORT NO
035D E60F             ANI 0FH
035F CDA803           CALL CVSST
0362 C608             ADI 08H      ADD DOT
0364 D304             OUT D8279
0366 78               MOV A,B
0367 0F               RRC
0368 0F               RRC
0369 0F               RRC
036A 0F               RRC
036B E60F             ANI 0FH
036D CDA803           CALL CVSST
0370 D304             OUT D8279
0372 3E37             MVI A,37H    ENTER CODE FOR P
0374 D304             OUT D8279
0376 AF               XRA A        CLEAR A
0377 D304             OUT D8279    CLEAR 2 LEFT DIGITS
0379 D304             OUT D8279
037B C9               RET
```

The I/O Port Write Function

```
037C AF        IWRST:XRA A        CLEAR A
037D 320E20           STA FTNFL    CLEAR FTN FLAG
0380 211820           LXI H,DDAT7  LOAD DATA ADDRESS
0383 7E               MOV A,M      LOAD MS ADD NIBBLE
0384 07               RLC
0385 07               RLC
0386 07               RLC
0387 07               RLC
0388 2B               DCX H
0389 86               ADD M        ADD LS NIBBLE
038A 47               MOV B,A
038B 2B               DCX H
038C 7E               MOV A,M      LOAD MS DATA NIBBLE
038D 07               RLC
038E 07               RLC
038F 07               RLC
0390 07               RLC
0391 2B               DCX H
0392 86               ADD M        ADD LS DATA NIBBLE
0393 4F               MOV C,A      B - PORT NO
                                   C - DATA
0394 211920           LXI H,IOSB0  LOAD START ADDRESS OF
                                   I/O ROUTINE SPACE
0397 3ED3             MVI A, 0D3H  I/O WRITE OPCODE
0399 77               MOV M,A
039A 23               INX H
039B 70               MOV M,B      LOAD PORT NO
```

```
039C 23           INX H
039D 3EC9         MVI A,0C9H   RETURN OPCODE
039F 77           MOV M,A
03A0 79           MOV A,C
03A1 CD1920       CALL IOSBO   WRITE DATA TO PORT
03A4 CD4103       CALL IDSST   UPDATE DISPLAY
03A7 C9           RET
```

Conversion from Hexadecimal to Seven Segment Code

```
                               HEX IN A REG CONVERTED
                               TO SEVEN SEGMENT CODE
03A8 E5     CVSST:PUSH H       SAVE INT REGISTERS
03A9 D5           PUSH D
03AA E60F         ANI 0FH      MASK FOR HEX
03AC 21B703       LXI H,CVSTB  LOAD STARTING ADDRESS OF
                               SEVEN SEGMENT TABLE
03AF 5F           MOV E,A      MOVE OFFSET TO E
03B0 1600         MVI D,00H    CLEAR D
03B2 19           DAD D        ADD OFFSET TO ADD IN HL
03B3 7E           MOV A,M      LOAD 7 SEG CODE
03B4 D1           POP D
03B5 E1           POP H
03B6 C9           RET
03B7 F360B5F4 CVSTB:DB 0F3H,60H,0B5H,0F4H
03BB 66D6D770       DB 66H,0D6H,0D7H,70H
03BF F7F677C7       DB 0F7H,0F6H,77H,0C7H
03C3 93E59717       DB 93H,0E5H,97H,17H
```

The Idle Loop

```
03C7 FB     LOPST:EI           ENABLE INTERRUPTS
03C8 3E40   LOP01:MVI A,40H    CLEAR SOD
03CA 30           SIM
03CB 3EC0         MVI A,0C0H   SET SOD
03CD 30           SIM
03CE C3C803       JMP LOP01    REPEAT LOOP TILL INTERRUPTED
```

The Data Block Move Routine

```
                                    BC - ADDRESS TO BE MOVED FROM
                                    DE - ADDRESS TO BE MOVED TO
                                    H - NO OF BYTES TO BE MOVED
03D1 F5       BKMST:PUSH PSW        SAVE ACCUMULATOR
03D2 0A       BKM01:LDAX B          LOAD DATA
03D3 12             STAX D          STORE IN NEW LOCN
03D4 03             INX B
03D5 13             INX D
03D6 25             DCR H           DECREMENT COUNTER
03D7 C2D203         JNZ BKM01       REPEAT UNTIL DONE
03DA F1             POP PSW         RESTORE ACCUMULATOR
03DB C9             RET
```

C. The Floating Point Arithmetic Package

Forwarding Addresses

```
0400 C31804 MULFA:JMP MULST    F.P. MULTIPLY
0403 C37E05 DIVFA:JMP DIVST    F.P. DIVIDE
0406 C38406 ADDFA:JMP ADDST    F.P. ADD
0409 C38307 SUBFA:JMP SUBST    F.P. SUBTRACT
040C C3FB04 NRMFA:JMP NRMST    F.P. NORMALIZE
040F C39E07 FSLFA:JMP FSLST    SHIFT LEFT
0412 C3B307 FSRFA:JMP FSRST    SHIFT RIGHT
0415 C3CD04 M16BF:JMP M16BS    16 BIT BINARY MULTIPLY
```

Floating Point Binary Multiplication

```
0418 F5      MULST:PUSH PSW    SAVE INT REGISTERS
0419 C5            PUSH B
041A D5            PUSH D
041B E5            PUSH H
041C AF            XRA A       CLEAR A
041D 322220        STA SIGNF   CLEAR SIGN FLAG FOR
                               POSITIVE RESULT
0420 3A1C20        LDA FPV11   LOAD MS MULTIPLIER BYTE
0423 47            MOV B,A     STORE
0424 17            RAL         SHIFT BIT 7 TO CY
0425 D23F04        JNC MUL01   SKIP IF NUMBER +VE
0428 3A2220        LDA SIGNF   LOAD SIGN FLAG
042B 2F            CMA         COMPLEMENT
042C 322220        STA SIGNF   STORE
042F 78            MOV A,B     LOAD MS MULTIPLIER
0430 2F            CMA         COMPLEMENT
0431 47            MOV B,A
0432 3A1D20        LDA FPV12   LOAD LS MULTIPLIER
0435 2F            CMA
0436 C601          ADI 01H     INCREMENT
0438 321D20        STA FPV12   STORE
```

```
043B 78           MOV A,B
043C CE00         ACI 00H       ADD CARRY TO C
043E 47           MOV B,A       STORE
043F 3A1D20 MUL01:LDA FPV12     LOAD LS MULTIPLIER
0442 4F           MOV C,A
0443 3A1F20       LDA FPV21     LOAD MS MULTIPLICAND BYTE
0446 57           MOV D,A
0447 17           RAL
0448 D26204       JNC MUL02     SKIP IF NO CARRY
044B 3A2220       LDA SIGNF
044E 2F           CMA           COMPLEMENT SIGN FLAG
044F 322220       STA SIGNF
0452 7A           MOV A,D       LOAD MS MULTIPLICAND
0453 2F           CMA
0454 57           MOV D,A
0455 3A2020       LDA FPV22     LOAD LS MULTIPLICAND
0458 2F           CMA
0459 C601         ADI 01H
045B 322020       STA FPV22
045E 7A           MOV A,D
045F CE00         ACI 00H
0461 57           MOV D,A
0462 3A2020 MUL02:LDA FPV22     LOAD LS MULTIPLICAND
0465 5F           MOV E,A
0466 CDCD04       CALL M16BS    MULTIPLY 16 BIT NUMBERS

                                STORE RESULTS
                                -------------
0469 44           MOV B,H       STORE HL
046A 4D           MOV C,L
046B A7           ANA A         CLEAR CARRY
046C 7B           MOV A,E       LOAD LS BYTE
046D 17           RAL           SHIFT TO MULTIPLY BY 2 TO
                                COMPENSATE FOR MANTISSA
                                MULTIPLICATION ALGORITHM'S
                                DIVISION BY TWO
046E 5F           MOV E,A
046F 7A           MOV A,D
0470 17           RAL
0471 57           MOV D,A
0472 79           MOV A,C
0473 17           RAL
0474 4F           MOV C,A
0475 78           MOV A,B
0476 17           RAL           SHIFT MS BYTE
0477 47           MOV B,A
0478 212320       LXI H,SHFT1
047B 70           MOV M,B       STORE MS PRODUCT BYTE
047C 23           INX H
047D 71           MOV M,C
047E 23           INX H
047F 72           MOV M,D
0480 23           INX H
0481 73           MOV M,E
```

```
0482 23             INX H        POINT HL TO SHFT5

                                 ADD EXPONENTS
                                 -------------
0483 3A1E20         LDA FPV13    LOAD MULTIPLIER EXPONENT
0486 47             MOV B,A      STORE
0487 17             RAL          SHIFT SIGN TO CY
0488 DAA404         JC MUL03     SKIP FOR -VE EXP
                                 TEST SECOND EXPONENT
048B 3A2120         LDA FPV23    LOAD MULTIPLICAND EXPONENT
048E 4F             MOV C,A
048F 17             RAL
0490 DAAC04         JC MUL04
                                 BOTH EXPONENTS +VE
0493 78             MOV A,B
0494 81             ADD C        ADD EXPONENTS
0495 322720         STA SHFT5    STORE RESULT
0498 17             RAL          SHIFT SIGN TO CY
0499 D2C504         JNC MUL06    JUMP TO MUL06 ROUTINE
                                 EXPONENT OVERFLOW IF CY SET
049C F5             PUSH PSW     SAVE REGISTERS
049D E5             PUSH H
049E 210066         LXI H,6600H  LOAD ERROR CODE 4
04A1 C34000         JMP ERRFA
04A4 3A2120  MUL03: LDA FPV23    TEST SECOND EXPONENT
04A7 4F             MOV C,A      SAVE MULTIPLICAND EXP
04A8 17             RAL          TEST SIGN
04A9 DAB404         JC MUL05     JUMP IF SECOND SIGN -VE
                                 ONE +VE AND ONE -VE SIGN
                                 NO EXP ERROR POSSIBLE
04AC 78      MUL04: MOV A,B
04AD 81             ADD C        ADD EXPONENTS
04AE 322720         STA SHFT5    STORE RESULT
04B1 C3C504         JMP MUL06
                                 BOTH EXP -VE
04B4 78      MUL05: MOV A,B
04B5 81             ADD C        ADD EXPONENTS
04B6 322720         STA SHFT5
04B9 17             RAL
04BA DAC504         JC MUL06     JUMP IF NO SIGN CHANGE
                                 IF CARRY CLEAR EXPONENT
                                 UNDERFLOW HAS OCCURED
04BD F5             PUSH PSW
04BE E5             PUSH H
04BF 2100D6         LXI H,0D600H LOAD ERROR CODE 5
04C2 C34000         JMP ERRFA
04C5 CDFB04  MUL06: CALL NRMST   NORMALIZE
04C8 E1             POP H
04C9 D1             POP D
04CA C1             POP B
04CB F1             POP PSW
04CC C9             RET
```

Sixteen Bit Binary Integer Multiplication

```
                                    16 BIT MULTIPLY
                                    BC - MULTIPLIER
                                    DE - MULTIPLICAND
04CD AF      M16BS:XRA A            CLEAR A, H, L
04CE 67            MOV H,A
04CF 6F            MOV L,A
04D0 E5            PUSH H           CLEAR TOP OF STACK
                                    HL - MS BYTES OF PRODUCT
                                    TOP OF STACK - LS BYTES
04D1 3E10          MVI A,10H
04D3 322820        STA FPLCT        STORE COUNTER
04D6 78      M16B1:MOV A,B          SHIFT MULTIPLIER RIGHT
04D7 1F            RAR              SHIFT RIGHT
04D8 47            MOV B,A
04D9 79            MOV A,C
04DA 1F            RAR
04DB 4F            MOV C,A
04DC D2E004        JNC M16B2        SKIP IF BIT 0
04DF 19            DAD D            ADD MULTIPLICAND TO MS HALF
                                    OF PRODUCT
04E0 7C      M16B2:MOV A,H          SHIFT PRODUCT RIGHT
04E1 A7            ANA A            CLEAR CARRY
04E2 1F            RAR              SHIFT RIGHT
04E3 67            MOV H,A          STORE
04E4 7D            MOV A,L          GET NEXT BYTE
04E5 1F            RAR
04E6 6F            MOV L,A
04E7 E3            XTHL             LOAD LS BYTES, STORE MS
04E8 7C            MOV A,H
04E9 1F            RAR
04EA 67            MOV H,A
04EB 7D            MOV A,L
04EC 1F            RAR
04ED 6F            MOV L,A
04EE E3            XTHL             STORE LS BYTES, LOAD MS
04EF 3A2820        LDA FPLCT        LOAD COUNTER
04F2 3D            DCR A            DECREMENT COUNTER
04F3 322820        STA FPLCT        STORE COUNTER
04F6 C2D604        JNZ M16B1
                                    MULTIPLICATION DONE
04F9 D1            POP D            LOAD LS PRODUCT BYTES
                                    DE - LS 2 BYTES OF PROD
                                    HL - MS 2 BYTES OF PROD
04FA C9            RET              EXIT
```

The Floating Point Normalization Routine

```
04FB F5        NRMST:PUSH PSW     SAVE INT REGISTERS
04FC C5              PUSH B
04FD D5              PUSH D
04FE E5              PUSH H
04FF 212320          LXI H,SHFT1
0502 46              MOV B,M      LOAD MANTISSA AND EXPONENT
0503 23              INX H        TO BE NORMALIZED
0504 4E              MOV C,M
0505 23              INX H
0506 56              MOV D,M
0507 23              INX H
0508 5E              MOV E,M
0509 23              INX H
050A 66              MOV H,M
050B 6C              MOV L,H      SAVE EXP TO TEST FOR
                                  SIGN ERROR
                                  TEST FOR ZERO MANTISSA
050C 78              MOV A,B
050D A7              ANA A        TEST
050E C22105          JNZ NRM01    SKIP IF NOT ZERO
0511 81              ADD C        TEST NEXT BYTE
0512 C22105          JNZ NRM01
0515 82              ADD D
0516 C22105          JNZ NRM01
0519 83              ADD E
051A C22105          JNZ NRM01    SKIP IF LAST BYTE NOT ZERO
051D 67              MOV H,A      SET EXPONENT TO ZERO
051E C37005          JMP NRM06    JUMP TO STORE RESULT
0521 78        NRM01:MOV A,B      LOAD MS BYTE
0522 17        NRM02:RAL          SHIFT SIGN BIT TO CY
0523 17              RAL          SHIFT MS DATA BIT TO CY
0524 DA3705          JC NRM03     SKIP IF BIT IS 1
                                  MS DATA BIT ZERO
0527 7B              MOV A,E      ROTATE ALL BYTES LEFT
0528 17              RAL
0529 5F              MOV E,A
052A 7A              MOV A,D
052B 17              RAL
052C 57              MOV D,A
052D 79              MOV A,C
052E 17              RAL
052F 4F              MOV C,A
0530 78              MOV A,B
0531 17              RAL
0532 47              MOV B,A
0533 25              DCR H        DECREMENT EXPONENT
0534 C32205          JMP NRM02    RETURN TO RETEST MS
                                  DATA BIT
0537 7A        NRM03:MOV A,D
0538 17              RAL          TEST FOR ROUNDUP
0539 D24805          JNC NRM04    SKIP IF NO ROUNDUP NEEDED
```

```
053C 0C           INR C        INCREMENT LS BYTE
053D C24805       JNZ NRM04    SKIP IF NO CARRY
0540 04           INR B        INCREMENT MS BYTE
0541 F24805       JP NRM04     SKIP IF MS BIT 0
0544 78           MOV A,B      CARRY INTO SIGN BIT HAS
                               OCCURRED
0545 0F           RRC          ROTATE RIGHT, NO NEED TO
                               ROTATE C AS IT MUST BE 0
0546 47           MOV B,A
0547 24           INR H        INCREMENT EXPONENT
                               TEST FOR EXPONENT UNDERFOLW
                               OVERFLOW NOT POSSIBLE
0548 7D     NRM04:MOV A,I      LOAD ORIGINAL EXPONENT
0549 17           RAL
054A D25D05       JNC NRM05    SKIP IF SIGN +VE
054D 7C           MOV A,H
054E 17           RAL          TEST FINAL EXPONENT
054F DA5D05       JC NRM05     SKIP, NO UNDERFLOW
0552 CD5D05       CALL NRM05   ERROR, FINISH ROUTINE FIRST
0555 F5           PUSH PSW     ERROR EXIT
0556 E5           PUSH H
0557 2100D7       LXI H,0D700H LOAD ERROR CODE 6
055A C34000       JMP ERRFA
                               ADD SIGN TO RESULT
055D 3A2220 NRM05:LDA SIGNF    LOAD SIGN FLAG
0560 A7           ANA A        TEST
0561 CA7005       JZ NRM06     SKIP IF 0, SIGN +VE
                               COMPLEMENT MANTISSA
0564 78           MOV A,B
0565 2F           CMA          COMPLEMENT
0566 47           MOV B,A
0567 79           MOV A,C
0568 2F           CMA          COMPLEMENT
0569 C601         ADI 01H      ADD CARRY
056B 4F           MOV C,A
056C 78           MOV A,B
056D CE00         ACI 00H
056F 47           MOV B,A
0570 54     NRM06:MOV D,H
0571 211F20       LXI H,FPV21  LOAD RESULT ADDRESS
0574 70           MOV M,B
0575 23           INX H
0576 71           MOV M,C
0577 23           INX H
0578 72           MOV M,D
0579 E1           POP H        RESTORE INT REGISTERS
057A D1           POP D
057B C1           POP B
057C F1           POP PSW
057D C9           RET
```

Floating Point Division

```
057E F5      DIVST:PUSH PSW      SAVE INT REGISTERS
057F C5            PUSH B
0580 D5            PUSH D
0581 E5            PUSH H
0582 AF            XRA A         CLEAR A
0583 322220        STA SIGNF     CLEAR SIGN FLAG FOR
                                 POSITIVE RESULT
0586 3A1C20        LDA FPV11     LOAD MS DIVIDEND BYTE
0589 47            MOV B,A       STORE
058A 17            RAL           SHIFT BIT 7 TO CY
058B D2A505        JNC DIV01     SKIP IF NUMBER +VE
058E 3A2220        LDA SIGNF     LOAD SIGN FLAG
0591 2F            CMA           COMPLEMENT
0592 322220        STA SIGNF     STORE
0595 78            MOV A,B       LOAD MS DIVIDEND
0596 2F            CMA           COMPLEMENT
0597 47            MOV B,A
0598 3A1D20        LDA FPV12     LOAD LS DIVIDEND
059B 2F            CMA
059C C601          ADI 01H       INCREMENT
059E 321D20        STA FPV12     STORE
05A1 78            MOV A,B
05A2 CE00          ACI 00H       ADD CARRY TO C
05A4 47            MOV B,A       STORE
05A5 3A1D20 DIV01:LDA FPV12      LOAD LS DIVIDEND
05A8 4F            MOV C,A
05A9 3A1F20        LDA FPV21     LOAD MS DIVISOR BYTE
05AC 57            MOV D,A
05AD 17            RAL
05AE D2C805        JNC DIV02     SKIP IF NO CARRY
05B1 3A2220        LDA SIGNF
05B4 2F            CMA           COMPLEMENT SIGN FLAG
05B5 322220        STA SIGNF
05B8 7A            MOV A,D       LOAD MS DIVISOR
05B9 2F            CMA
05BA 57            MOV D,A
05BB 3A2020        LDA FPV22     LOAD LS DIVISOR
05BE 2F            CMA
05BF C601          ADI 01H
05C1 322020        STA FPV22
05C4 7A            MOV A,D
05C5 CE00          ACI 00H
05C7 57            MOV D,A
05C8 3A2020 DIV02:LDA FPV22      LOAD LS DIVIDER
05CB 5F            MOV E,A
                                 BC - DIVIDEND MANTISSA
                                 DE - DIVISOR MANTISSA
                                 TEST FOR 0 DIVISOR
05CC AF            XRA A         CLEAR A
05CD 67            MOV H,A       CLEAR STACK TOP FOR LATER USE
05CE 6F            MOV L,A
```

```
05CF E5           PUSH H
05D0 82           ADD D        ADD MS DIVISOR BYTE
05D1 C2E005       JNZ DIV03    SKIP  IF NOT A ZERO
05D4 83           ADD E        ADD LS DIVISOR BYTE
05D5 C2E005       JNZ DIV03    SKIP IF NOT ZERO
05D8 F5           PUSH PSW     ERROR EXIT
05D9 E5           PUSH H
05DA 2100F7       LXI H,0F700H LOAD ERROR CODE 8
05DD C34000       JMP ERRFA

                               DIVIDE MANTISSAS
                               ----------------
05E0 260F   DIV03:MVI H,0FH    LOAD COUNTER
                               DIVIDE LOOP
05E2 79     DIV04:MOV A,C      LOAD LS DIVIDEND
05E3 93           SUB E        SUB LS DIVISOR
05E4 6F           MOV L,A      STORE RESULT
05E5 78           MOV A,B      LOAD MS DIVIDEND
05E6 9A           SBB D        SUB MS DEVISOR
05E7 DAF805       JC DIV05     SKIP IF NEGATIVE RESULT
                               POS RESULT, REPLACE DIVIDEND
                               WITH REMAINDER
05EA 47           MOV B,A      LOAD MS DIVIDEND
05EB 4D           MOV C,L      LOAD LS DIVIDEND
05EC E3           XTHL         SWAP HL WITH STACK TOP
05ED 37           STC          SET CARRY
05EE 7D           MOV A,L      LOAD LS QUOTIENT
05EF 17           RAL          ROTATE CARRY IN
05F0 6F           MOV L,A
05F1 7C           MOV A,H      LOAD MS QUOTIENT
05F2 17           RAL
05F3 67           MOV H,A
05F4 E3           XTHL         SWAP HL, WITH STACK
05F5 C30106       JMP DIV06
05F8 E3     DIV05:XTHL         NEG RESULT
05F9 7D           MOV A,L      LOAD LS QUOTIENT
05FA A7           ANA A        CLEAR CARRY
05FB 17           RAL          SHIFT IN 0
05FC 6F           MOV L,A
05FD 7C           MOV A,H      LOAD MS QUOTIENT
05FE 17           RAL
05FF 67           MOV H,A
0600 E3           XTHL
                               ROTATE DIVIDEND
0601 79     DIV06:MOV A,C      LOAD LS DIVIDEND
0602 A7           ANA A        CLEAR CARRY
0603 17           RAL          ROTATE
0604 4F           MOV C,A
0605 78           MOV A,B      LOAD MS DIVIDEND
0606 17           RAL
0607 47           MOV B,A
0608 25           DCR H        DECREMENT COUNTER
0609 C2E205       JNZ DIV04    REPEAT LOOP IF NOT ZERO
                               DIVISION COMPLETE
```

```
060C 79                MOV A,C       SUBTRACT DIVISOR FROM
060D 93                SUB E         DIVIDEND TO TEST
060E 78                MOV A,B       FOR RCUNDUP
060F 9A                SBB D
0610 DA1D06            JC DIV07      RESULT NEG, NO ROUNDUP

                                     ADD ROUNDUP
                                     -----------
0613 E1                POP H         LOAD QUOTIENT
0614 7D                MOV A,L
0615 C601              ADI 01H        INCREMENT LS BYTE
0617 6F                MOV L,A
0618 7C                MOV A,H       LOAD MS BYTE
0619 CE00              ACI 00H       ADD CARRY
061B 67                MOV H,A
061C E5                PUSH H        PUSH TO COMPENSATE FOR
                                     FOLLOWING POP
061D D1          DIV07:POP D         LOAD QUOTIENT
061E 212320            LXI H,SHFT1   LOAD ADD OF SHIFT LOCATIONS
0621 72                MOV M,D       STORE QUOTIENT MANTISSA
0622 23                INX H
0623 73                MOV M,E
0624 AF                XRA A         CLEAR A
0625 23                INX H
0626 77                MOV M,A
0627 23                INX H
0628 77                MOV M,A
0629 23                INX H         POINT H AT SHFT5

                                     SUBTRACT EXPONENTS
                                     ------------------
062A 3A1E20            LDA FPV13     LOAD DIVIDEND EXPONENT
062D 47                MOV B,A       STORE
                                     INCREMENT EXP TO COMPENSATE
                                     FOR EXTRA LEADING ZERO
062E 17                RAL           SHIFT SIGN TO CY
062F D23606            JNC DIV08     SKIP FOR +VE EXP
0632 04                INR B         INCREMENT DIVIDEND EXP
0633 C35006            JMP DIV10     GO TC DIVISOR EXP TEST
0636 04          DIV08:INR B         INCREMENT EXPONENT
0637 F24206            JP DIV09      IF OVERFLOW INTO SIGN BIT
                                     DID NOT OCCURR
                                     ERROR EXIT FOR EXP OVERFLOW
063A F5                PUSH PSW
063B E5                PUSH H
063C 21000E            LXI H,0E00H   LOAD ERROR CODE 7
063F C34000            JMP ERRFA
                                     TEST SECOND EXPONENT
0642 3A2120      DIV09:LDA FPV23     LOAD DIVISOR EXPONENT
0645 4F                MOV C,A
0646 17                RAL
0647 DA5E06            JC DIV11
                                     BOTH EXPONENTS +VE
064A 78                MOV A,B
```

```
064B 91           SUB C        SUB DIVISOR EXP
064C 77           MOV M,A      STORE RESULT IN SHIFT5
064D C37C06       JMP DIV13    JUMP TO DIV13 ROUTINE
0650 3A2120 DIV10:LDA FPV23    TEST SECOND EXPONENT
0653 4F           MOV C,A      SAVE DIVISOR EXP
0654 17           RAL          TEST SIGN
0655 D26D06       JNC DIV12    JUMP IF SECOND SIGN +VE
                               TWO - SIGNS
0658 78           MOV A,B
0659 91           SUB C        SUB DIVISOR EXPONENT
065A 77           MOV M,A      STORE EXP
065B C37C06       JMP DIV13
                               DIVIDEND EXP +VE AND
                               DIVISOR EXP -VE
065E 78     DIV11:MOV A,B
065F 91           SUB C        SUBTRACT
0660 77           MOV M,A      STORE RESULT
0661 17           RAL
0662 D27C06       JNC DIV13    SKIP IF NO OVERFLOW
                               INTO SIGN BIT
0665 F5           PUSH PSW     ERROR EXIT
0666 E5           PUSH H
0667 2100F6       LXI H,0F600H LOAD ERROR CODE 9
066A C34000       JMP ERRFA
                               DIVIDEND EXP -VE AND
                               DEVISOR EXP +VE
066D 78     DIV12:MOV A,B
066E 91           SUB C        SUBTRACT
066F 77           MOV M,A      STORE RESULT
0670 17           RAL
0671 DA7C06       JC DIV13     SKIP IF SIGN BIT NOT CHANGED
0674 F5           PUSH PSW     ERROR EXIT
0675 E5           PUSH H
0676 21F360       LXI H,60F3H  LOAD ERROR CODE 10
0679 C34000       JMP ERRFA
067C E1     DIV13:POP H        RESTORE INT REGISTERS
067D D1           POP D
067E C1           POP B
067F F1           POP PSW
0680 CDFB04       CALL NRMST   NORMALIZE
0683 C9           RET
```

Floating Point Addition

```
0684 F5      ADDST:PUSH PSW      SAVE INTERNAL REGISTERS
0685 C5            PUSH B
0686 D5            PUSH D
0687 E5            PUSH H
0688 3A1E20        LDA FPV13     LOAD FIRST EXPONENT
068B 47            MOV B,A       STORE
068C 3A2120        LDA FPV23     LOAD SECOND EXPONENT
068F 4F            MOV C,A       STORE
0690 90            SUB B         SUB 1ST EXPONENT FROM 2ND
0691 57            MOV D,A       SAVE DIFFERENCE
0692 FAE006        JM ADD06      EXPONENT 1 IS LARGER, JUMP
                                 TO PROPER ROUTINE, ELSE
                                 EXP 2 GT OR EQU TO EXP 1
0695 DE10          SBI 10H       SUBTRACT 16D
0697 FA9D06        JM ADD01      SKIP IS EXPS WITHIN RANGE,
                                 ELSE LEAVE LARGER NUMBER
                                 IN RESULT FILE
069A C37E07        JMP ADD14     JUMP TO END
069D 59      ADD01:MOV E,C       SAVE LARGER EXPONENT
069E 3A1C20        LDA FPV11     LOAD MS BYTE OF SMALLER NO
06A1 67            MOV H,A
06A2 3A1D20        LDA FPV12     LOAD LS BYTE
06A5 6F            MOV L,A
06A6 E5            PUSH H        SAVE MS WORKSPACE HALF
06A7 AF            XRA A         CLEAR A
06A8 67            MOV H,A       CLEAR LS WORKSPACE HALF
06A9 6F            MOV L,A
06AA E3            XTHL          LOAD MS WORKSPACE HALF
06AB 14            INR D         INCREMENT EXP DIFFERENCE
                                 TO GET EXTRA SHIFT TO
                                 LEAVE ROOM FOR OVERFLOW BIT
06AC 7C            MOV A,H       LOAD MS MANTISSA BYTE
06AD A7            ANA A         TEST
06AE FABC06        JM ADD03      SKIP IF SIGN -VE ELSE DO

                                 +VE MANT NORMALIZATION
                                 ----------------------
06B1 A7      ADD02:ANA A         CLEAR CARRY
06B2 CDB307        CALL FSRST    CALL SHIFT ROUTINE
06B5 15            DCR D         DECREMENT COUNTER
06B6 C2B106        JNZ ADD02     REPEAT UNTIL COUNTER 0
06B9 C3C406        JMP ADD04     LOAD AND SHIFT OTHER MANTISSA

                                 -VE MANT NORMALIZATION
                                 ----------------------
06BC 37      ADD03:STC           SET CARRY
06BD CDB307        CALL FSRST    CALL SHIFT ROUTINE
06C0 15            DCR D         DECREMENT COUNTER
06C1 C2BC06        JNZ ADD03     REPEAT LOOP UNTIL COUNTER 0
06C4 E5      ADD04:PUSH H        PUSH MS WORKSPACE HALF ON TOP
                                 OF LS HALF
```

```
06C5 D5                PUSH D       SAVE LARGER EXP
06C6 D5                PUSH D       STACK TOP BECOMES LS HALF
                                    OF SECOND NUMBER
06C7 3A2020            LDA FPV22    LOAD LS BYTE OF MANTISSA
                                    OF LARGER NUMBER
06CA 6F                MOV L,A
06CB 3A1F20            LDA FPV21    LOAD MS BYTE
06CE 67                MOV H,A
06CF A7                ANA A        TEST, CLEAR CARRY
06D0 FAD906            JM ADD05     SKIP IF MANTISSA IS NEG
                                    ELSE SHIFT RIGHT
06D3 CDB307            CALL FSRST   SHIFT RIGHT
06D6 C33307            JMP ADD12    JUMP TO ADDITION OF MANTISSA
06D9 37         ADD05: STC          SET CARRY
06DA CDB307            CALL FSRST   SHIFT IN A 1
06DD C33307            JMP ADD12    JUMP TO ADDITION OF MANTISSAS

                                    EXPONENT 1 IS LARGER
                                    --------------------

06E0 C610       ADD06: ADI 10H      ADD 16
06E2 F2F306            JP ADD07     SKIP IF EXP IN RANGE
                                    ELSE LOAD LARGER NO TO RESULT
06E5 011C20            LXI B,FPV11  LOAD BLOCKMOVE DATA
06E8 111F20            LXI D,FPV21
06EB 2603              MVI H,03H
06ED CD4F00            CALL BKMFA   MOVE DATA
06F0 C37E07            JMP ADD14    JUMP TO END
06F3 58         ADD07: MOV E,B      SAVE LARGER EXP
06F4 3A1F20            LDA FPV21    LOAD MS BYTE
06F7 67                MOV H,A
06F8 3A2020            LDA FPV22    LOAD LS BYTE
06FB 6F                MOV L,A
06FC E5                PUSH H       SAVE MS WORKSPACE HALF
06FD AF                XRA A        CLEAR A,H,L
06FE 67                MOV H,A
06FF 6F                MOV L,A
0700 E3                XTHL         STORE LS WORKSPACE HALF
                                    AND LOAD MS HALF
0701 15                DCR D        DECREMENT -VE DIFF TO GET
                                    AN EXTRA SHIFT
0702 7C                MOV A,H      LOAD MS MANTISSA BYTE
0703 A7                ANA A        TEST
0704 FA1207            JM ADD09     SKIP IF -VE ELSE

                                    POS MANTISSA NORMALIZATION
                                    --------------------------

0707 A7         ADD08: ANA A        CLEAR CY
0708 CDB307            CALL FSRST   CALL NORMALIZATION ROUTINE
070B 14                INR D        INCREMENT COUNTER
070C C20707            JNZ ADD08    REPEAT UNTIL COUNTER 0
070F C31A07            JMP ADD10
```

```
                                   -VE EXP NORMALIZATION
                                   ---------------------
0712 37        ADD09:STC           SET CARRY
0713 CDB307          CALL FSRST    SHIFT IN A 1
0716 14              INR D         INCREMENT COUNTER
0717 C21207          JNZ ADD09     REPEAT UNTIL COUNTER 0
071A E5        ADD10:PUSH H        PUSH MS WORKSPACE HALF
071B D5              PUSH D        SAVE LARGER EXPONENT
071C D5              PUSH D        STACK TOP BECOMES LS HALF
                                   OF SECOND NUMBER
071D 3A1D20          LDA FPV12     LOAD LS BYTE OF MANTISSA OF
                                   LARGER NUMBER
0720 6F              MOV L,A
0721 3A1C20          LDA FPV11     LOAD MS BYTE
0724 67              MOV H,A
0725 A7              ANA A         TEST, CLEAR CY
0726 FA2F07          JM ADD11      SKIP IF MANTISSA NEG,
                                   ELSE SHIFT RIGHT
0729 CDB307          CALL FSRST    SHIFT RIGHT
072C C33307          JMP ADD12     JUMP TO MANTISSA ADDITION

                                   SHIFT RIGHT FOR -VE MANTISSA
                                   ----------------------------
072F 37        ADD11:STC           SET CARRY
0730 CDB307          CALL FSRST    SHIFT IN A 1

                                   ADDITION OF MANTISSAS
                                   ---------------------
                                   E - LARGER EXPONENT
                                   HL - MS HALF OF MANTISSA
                                   OF LARGER NUMBER
                                   STACK TOP - LS BYTE
                                   S.T. + 2 - LARGER EXPONENT
                                   S. T. + 4 - MS HALF OF MANT
                                   OF SMALLER NUMBER
                                   S. T. + 6 - LS HALF OF  MANT
0733 44        ADD12:MOV B,H       MOVE 2 MS BYTES OF MANTISSA
                                   OF LARGER NUMBER
0734 4D              MOV C,L
0735 E1              POP H         POP LS MANTISSA BYTE OF
                                   LARGER NUMBER
0736 54              MOV D,H       MOVE TO D
0737 E1              POP H         LOAD LARGER EXPONENT
0738 5D              MOV E,L       STORE
0739 E1              POP H         POP MS HALF OF MANTISSA OF
                                   SMALLER NUMBER
073A E3              XTHL          LOAD LS HALF OF WORK SPACE
073B 7C              MOV A,H       LOAD 3RD BYTE
073C 82              ADD D
073D 67              MOV H,A
073E E3              XTHL          LOAD MS HALF OF WORK SPACE
073F 7D              MOV A,L       LOAD AND ADD 2ND BYTE
0740 89              ADC C
0741 4F              MOV C,A
```

```
0742 7C              MOV A,H      LOAD AND ADD MS BYTE
0743 88              ADC B
0744 47              MOV B,A
0745 7B              MOV A,E      LOAD EXPONENT
0746 D1              POP D        LOAD LS 2 BYTES TO DE
0747 212720          LXI H,SHFT5  SET POINTER TO NORMALIZATION
                                  ROUTINE DATA FILE
074A 3C              INR A        INCREMENT EXPONENT TO
                                  COMPENSATE FOR EXTRA SHIFT
074B 77              MOV M,A      STORE FOR NORMALIZATION
074C AF              XRA A        CLEAR A
074D 322220          STA SIGNF    CLEAR SIGN FLAG
0750 78              MOV A,B      LOAD MS MANTISSA BYTE
0751 A7              ANA A        TEST
0752 F27307          JP ADD13     SKIP NEXT LINES IF MATISSA
                                  POSITIVE, ELSE

                                  COMPLEMENT MANT, SET SIGNF
                                  --------------------------
0755 2F              CMA          COMPLEMENT MS BYTE
0756 47              MOV B,A
0757 79              MOV A,C
0758 2F              CMA          COMPLEMENT 2ND BYTE
0759 4F              MOV C,A
075A 7A              MOV A,D
075B 2F              CMA          COMPLEMENT 3RD BYTE
075C 57              MOV D,A
075D 7B              MOV A,E
075E 2F              CMA          COMPLEMENT LS BYTE
075F C601            ADI 01H      INCREMENT LS BYTE
0761 5F              MOV E,A
0762 7A              MOV A,D
0763 CE00            ACI 00       ADD CARRY
0765 57              MOV D,A
0766 79              MOV A,C
0767 CE00            ACI 00H      ADD CY TO 2ND BYTE
0769 4F              MOV C,A
076A 78              MOV A,B
076B CE00            ACI 00H      ADD CY TO MS BYTE
076D 47              MOV B,A
076E 3EFF            MVI A,0FFH
0770 322220          STA SIGNF    SET SIGN FLAG FOR - SIGN
0773 2B       ADD13: DCX H
0774 73              MOV M,E      STORE LS BYTE
0775 2B              DCX H
0776 72              MOV M,D
0777 2B              DCX H
0778 71              MOV M,C
0779 2B              DCX H
077A 70              MOV M,B      STORE LS BYTE
077B CDFB04          CALL NRMST   NORMALIZE
077E E1       ADD14: POP H        RESTORE INT REGISTERS
077F D1              POP D
0780 C1              POP B
```

```
0781 F1             POP PSW
0782 C9             RET           DONE
```

Floating Point Subtraction

```
0783 F5       SUBST:PUSH PSW      SAVE INTERNAL REGISTERS
0784 E5             PUSH H
0785 C5             PUSH B

                                  COMPLEMENT  MINUEND
                                  -------------------
0786 211F20         LXI H,FPV21   POINT TO MINUEND
0789 7E             MOV A,M       LOAD MS MANTISSA BYTE
078A 2F             CMA           COMPLEMENT
078B 47             MOV B,A
078C 23             INX H
078D 7E             MOV A,M       LOAD LS MANTISSA BYTE
078E 2F             CMA           COMPLEMENT
078F C601           ADI 01H       INCREMENT
0791 77             MOV M,A       STORE IN MEMORY
0792 78             MOV A,B       LOAD COMPLEMENTED MS BYTE
0793 CE00           ACI 00H       ADD CARRY
0795 2B             DCX H
0796 77             MOV M,A       STORE IN MEMORY
0797 CD8406         CALL ADDST    ADD INVERTED MINUEND TO
                                  SUBTRAHEND
079A C1             POP B         RESTORE INTERNAL REGISTERS
079B E1             POP H
079C F1             POP PSW
079D C9             RET           DONE
```

Four Byte Shift Left

```
                                  HL -MS 2 BYTES
                                  STACK TOP - LS 2 BYTES
                                  CY - VALUE TO BE SHIFTED IN
079E F3       FSLST:DI            DISABLE INTERRUPTS TO
                                  PREVENT ERRORS DUE TO
                                  STACK MANIPULATION
079F 33             INX SP        MOVE STACK POINTER
07A0 33             INX SP        TO STORED DATA
07A1 E3             XTHL          LOAD LS WORKSPACE HALF
07A2 7D             MOV A,L
07A3 17             RAL           SHIFT LS BYTE
```

```
07A4 6F         MOV L,A
07A5 7C         MOV A,H
07A6 17         RAL         SHIFT 3RD BYTE
07A7 67         MOV H,A
07A8 E3         XTHL        LOAD MS WORKSPACE HALF
07A9 7D         MOV A,L
07AA 17         RAL         SHIFT 2ND BYTE
07AB 6F         MOV L,A
07AC 7C         MOV A,H
07AD 17         RAL         SHIFT MS BYTE
07AE 67         MOV H,A
07AF 3B         DCX SP      RESTORE STACK POINTER
07B0 3B         DCX SP      TO RETURN ADDRESS
07B1 FB         EI          RENABLE INTERRUPTS
07B2 C9         RET         DONE
```

<u>Four Byte Shift Right</u>

```
                            SIMILAR TO 4 BYTE SHIFT LEFT
07B3 F3   FSRST:DI
07B4 33         INX SP
07B5 33         INX SP
07B6 7C         MOV A,H
07B7 1F         RAR         SHIFT MS BYTE
07B8 67         MOV H,A
07B9 7D         MOV A,L
07BA 1F         RAR         SHIFT 2ND BYTE
07BB 6F         MOV L,A
07BC E3         XTHL        LOAD LS WORKSPACE HALF
07BD 7C         MOV A,H
07BE 1F         RAR         SHIFT 3RD BYTE
07BF 67         MOV H,A
07C0 7D         MOV A,L
07C1 1F         RAR         SHIFT LS BYTE
07C2 6F         MOV L,A
07C3 E3         XTHL        LOAD MS WORKSPACE HALF
07C4 3B         DCX SP
07C5 3B         DCX SP
07C6 FB         EI
07C7 C9         RET
```

Forwarding Addresses

```
0800 C30608 DTBFA:JMP DTBST
0803 C3DA08 BTDFA:JMP BTDST
```

Floating Point Decimal to Binary Conversion

```
0806 F5     DTBST:PUSH PSW     SAVE INTERNAL REGISTERS
0807 C5           PUSH B
0808 D5           PUSH D
0809 E5           PUSH H
080A AF           XRA A        CLEAR A,H,L,C
080B 67           MOV H,A
080C 6F           MOV L,A
080D 4F           MOV C,A
080E 0603         MVI B,03H    LOAD COUNTER
0810 C5           PUSH B       STORE COUNTER
0811 112920       LXI D,DECV1  SET POINTER TO START OF
                               DECIMAL NUMBER FILE
0814 322220       STA SIGNF    CLEAR SIGN FLAG
0817 1A           LDAX D       LOAD SIGN BYTE
0818 13           INX D        INCREMENT POINTER
0819 A7           ANA A        TEST
081A CA2208       JZ DTB01     SKIP IF SIGN +VE
081D 3EFF         MVI A,0FFH   LOAD CODE FOR - SIGN
081F 322220       STA SIGNF    STORE SIGN FLAG

                               CONVERT MANTISSA
                               ----------------
0822 1A     DTB01:LDAX D       LOAD DIGIT
0823 13           INX D        INCREMENT POINTER
0824 85           ADD L        ADD L
0825 6F           MOV L,A
0826 79           MOV A,C      CLEAR A
0827 8C           ADC H        ADD H, CARRY IS CLEAR
0828 67           MOV H,A
0829 44           MOV B,H      STORE
082A 4D           MOV C,L
082B 7D           MOV A,L
082C 17           RAL          SHIFT LEFT
082D 6F           MOV L,A
082E 7C           MOV A,H
082F 17           RAL
0830 67           MOV H,A      HL NOW MULT BY 2
0831 7D           MOV A,L
0832 17           RAL          SHIFT AGAIN
0833 6F           MOV L,A
0834 7C           MOV A,H
```

```
0835 17          RAL
0836 67          MOV H,A       HL NOW MULT BY 4
0837 09          DAD B         ADD B, NOW X 5
0838 7D          MOV A,L       SHIFT LEFT AGAIN
0839 17          RAL
083A 6F          MOV L,A
083B 7C          MOV A,H
083C 17          RAL
083D 67          MOV H,A       HL NOW X 10
083E C1          POP B         LOAD COUNTER, CLEAR C
083F 05          DCR B         DECREMENT COUNTER
0840 C5          PUSH B        SAVE DECREMENTED COUNTER
0841 C22208      JNZ DTB01     REPEAT LOOP IF
                               COUNTER NOT ZERO

0844 C1          POP B         CLEAR COUNTER FROM STACK
0845 1A          LDAX D        LOAD LAST DIGIT
0846 13          INX D         INCREMENT POINTER
0847 85          ADD L         ADD TO L
0848 6F          MOV L,A
0849 79          MOV A,C       CLEAR A
084A 8C          ADC H         ADD CY TO H
084B 67          MOV H,A       BINARY MANTISSA NOW IN HL
084C E5          PUSH H        STORE DATA
084D 212720      LXI H,SHFT5   LOAD DATA ADDRESS
                               FOR SHIFT ROUTINE

0850 360F        MVI M,0FH     LOAD EXP TO COMPENSATE FOR
                               POSN OF BINARY POINT, 15D

0852 2B          DCX H
0853 71          MOV M,C       CLEAR LS BYTE
0854 2B          DCX H
0855 71          MOV M,C       CLEAR NEXT BYTE
0856 2B          DCX H
0857 C1          POP B         RELOAD DATA
0858 71          MOV M,C       STORE NEXT DATA BYTE
0859 2B          DCX H
085A 70          MOV M,B       LOAD MS DATA BYTE
085B CD0C04      CALL NRMFA    CALL NORMALIZATION ROUTINE
                               BINARY RESULT NOW IN FPV2N
                               MULT BY .001 TO COMPENSATE
                               FOR MOVED DECIMAL POINT
085E 211C20      LXI H,FPV11   SET DATA POINTER
0861 3641        MVI M,41H     LOAD BINARY .001
0863 23          INX H
0864 3689        MVI M,89H
0866 23          INX H
0867 36F7        MVI M,0F7H
0869 CD0004      CALL MULFA    MULTIPLY BY .001
086C EB          XCHG          MOVE DEC DATA POINTER TO HL
086D 7E          MOV A,M       LOAD EXPONENT SIGN FLAG
086E 23          INX H
086F A7          ANA A         TEST
0870 C2A408      JNZ DTB05     SKIP TO - EXPONENT
                               NORMALIZATION ROUTINE,
                               ELSE DO
```

```
                                      + SIGN NORMALIZATION ROUTINE
                                      ----------------------------
0873 46               MOV B,M         LOAD MS EXPONENT DIGIT
0874 23               INX H
0875 4E               MOV C,M         LOAD LS EXPONENT DIGIT
                                      EXPONENT STORED IN BC
0876 211C20           LXI H,FPV11     LOAD MULTIPLIER ADDRESS
0879 3650             MVI M,50H       LOAD BINARY 10
087B 23               INX H
087C 3600             MVI M,00H
087E 23               INX H
087F 3604             MVI M,04H

                                      COMPENSATE FOR LOW ORDER
                                      ------------------------
                                      EXPONENT DIGIT
                                      --------------
0881 79        DTB02:MOV A,C          LOAD LS DIGIT
0882 A7               ANA A           TEST
0883 CA8D08           JZ DTB03        SKIP TO NEXT DIGIT IF ZERO
0886 0D               DCR C           DECREMENT EXPONENT
0887 CD0004           CALL MULFA      MULTIPLY FPV2N BY 10
088A C38108           JMP DTB02       RETURN AND REPEAT LOOP

                                      COMPENSATE FOR HIGH ORDER
                                      -------------------------
                                      EXPONENT DIGIT
                                      --------------
088D 211C20    DTB03:LXI H,FPV11      LOAD MULTIPLIER ADDRESS
0890 364A             MVI M,4AH       LOAD BINARY 1E10
0892 23               INX H
0893 3681             MVI M,81H
0895 23               INX H
0896 3622             MVI M,22H
0898 78        DTB04:MOV A,B          LOAD MS DIGIT
0899 A7               ANA A           TEST
089A CAD508           JZ DTB09        DONE, GO TO END
089D 05               DCR B           DECREMENT EXPONENT DIGIT
089E CD0004           CALL MULFA      MULT FPV2N BY 100
08A1 C39808           JMP DTB04       RETURN AND REPEAT LOOP

                                      -VE EXPONENT NORMALIZATION
                                      --------------------------
08A4 46        DTB05:MOV B,M          LOAD MS EXPONENT DIGIT
08A5 23               INX H
08A6 4E               MOV C,M         LOAD LS EXPONENT DIGIT
08A7 211C20           LXI H,FPV11     LOAD DIVISOR ADDRESS
08AA 3666             MVI M,66H       LOAD WITH BINARY .10
08AC 23               INX H
08AD 3666             MVI M,66H
08AF 23               INX H
08B0 36FD             MVI M,0FDH
08B2 79        DTB06:MOV A,C          LOAD LS DIGIT
```

```
08B3 A7                 ANA A          TEST
08B4 CABE08             JZ DTB07       DIGIT ZERO, GO TO ROUTINE TO
                                       NORMALIZE NEXT DIGIT
08B7 0D                 DCR C          DECREMENT EXPONENT
08B8 CD0004             CALL MULFA     MULTIPLY BY .10
08BB C3B208             JMP DTB06      REPEAT LOOP
08BE 211C20      DTB07:LXI H,FPV11     LOAD DIVISOR ADDRESS
08C1 366D               MVI M,6DH      LOAD WITH BINARY 1E-10
08C3 23                 INX H
08C4 36F4               MVI M,0F4H
08C6 23                 INX H
08C7 36DF               MVI M,0DFH
08C9 78          DTB08:MOV A,B         LOAD MS DIGIT
08CA A7                 ANA A          TEST
08CB CAD508             JZ DTB09       DONE, GO TO END
08CE 05                 DCR B          DECREMENT EXPONENT DIGIT
08CF CD0004             CALL MULFA     MULTIPLY FPV2N BY 1E-10
08D2 C3C908             JMP DTB08      REPEAT LOOP

                                       END
                                       ---
                                       BINARY VALUE IN FPV2N
08D5 E1          DTB09:POP H           RETURN INTERNAL REGISTERS
08D6 D1                 POP D
08D7 C1                 POP B
08D8 F1                 POP PSW
08D9 C9                 RET            DONE
```

<u>Floating Point Binary to Decimal Conversion</u>

```
08DA F5          BTDST:PUSH PSW        SAVE INTERNAL REGISTERS
08DB C5                 PUSH B
08DC D5                 PUSH D
08DD E5                 PUSH H

                                       TEST FOR ZERO MANTISSA
                                       ----------------------

08DE 3A1F20             LDA FPV21      LOAD MS MANT BYTE
08E1 A7                 ANA A          TEST
08E2 C2F408             JNZ BTD02      NOT ZERO, CONVERT, ELSE
                                       OUTPUT DECIMAL 0
08E5 212920             LXI H,DECV1    DESTINATION POINTER
08E8 0608               MVI B,08H      LOAD COUNTER
08EA AF                 XRA A          CLEAR A
08EB 77          BTD01:MOV M,A         CLEAR DECIMAL DATA
08EC 23                 INX H
08ED 05                 DCR B          DECREMENT COUNTER
08EE C2EB08             JNZ BTD01      REPEAT
08F1 C34A0A             JMP BTD0E      GO TO END
```

```
08F4 AF      BTD02:XRA A        CLEAR A
08F5 5F            MOV E,A      STORE 0 DECIMAL EXPONENT
08F6 212120        LXI H,FPV23  POINT TO EXPONENT
08F9 7E            MOV A,M      LOAD
08FA 3C            INR A        MULT DATA BY 2 TO PROVIDE
                                EXTRA SHIFT LEFT TO
                                COMPENSATE FOR SIGN BIT
                                WHEN SHIFTING OUT DECIMAL
                                DIGITS
08FB 77            MOV M,A      STORE NEW EXPONENT
08FC A7            ANA A        TEST
08FD FA2209        JM BTD04     SKIP TO ROUTINE FOR
                                -VE EXPONENT NORMALIZATION

                                + EXPONENT NORMALIZATION
                                --------------------------
0900 211C20        LXI H,FPV11  LOAD POINTER
0903 3666          MVI M,66H    LOAD BINARY .10
0905 23            INX H
0906 3666          MVI M,66H
0908 23            INX H
0909 36FD          MVI M,0FDH
090B 212120        LXI H,FPV23  SET POINTER TO EXP
                                OF NUMBER BEING CONVERTED
090E CD0004 BTD03:CALL MULFA    MULTIPLY BY .10
0911 1C            INR E        INCREMENT DECIMAL EXPONENT
0912 7E            MOV A,M      LOAD NEW EXPONENT
0913 A7            ANA A        TEST
0914 F20E09        JP BTD03     REPEAT LOOP
0917 211F20        LXI H,FPV21  LOAD DATA POINTER
091A 46            MOV B,M      LOAD BINARY MANTISSA
091B 23            INX H
091C 4E            MOV C,M
091D 23            INX H
091E 6E            MOV L,M      LOAD BINARY EXPONENT
091F C34809        JMP BTD06    NEXT NORMALIZATION STAGE

                                - EXPONENT NORMALIZATION
                                ------------------------
0922 211C20 BTD04:LXI H,FPV11   LOAD POINTER
0925 3650          MVI M,50H    LOAD BINARY 10
0927 23            INX H
0928 3600          MVI M,00H
092A 23            INX H
092B 3604          MVI M,04H
092D 211F20 BTD05:LXI H,FPV21   LOAD DATA POINTER
0930 46            MOV B,M      STORE DATA BEFORE MULTIPLYING
0931 23            INX H
0932 4E            MOV C,M
0933 23            INX H
0934 56            MOV D,M
0935 CD0004        CALL MULFA   MULT BY 10
0938 1D            DCR E        DECR DECIMAL EXPONENT
0939 7E            MOV A,M      LOAD NEW BINARY EXPONENT
```

```
093A A7            ANA A        TEST
093B FA2D09        JM BTD05     REPEAT LOOP IF EXP SITLL -VE
                                EXPONENT NOW +VE
093E 1C            INR E        INCR DECIMAL EXP TO
                                CANCEL LAST MULTIPLY
093F 211F20        LXI H,FPV21  LOAD DATA ADDRESS POINTER
0942 70            MOV M,B      RESTORE VALUES BEFORE
                                LAST MULTIPLICATION
0943 23            INX H
0944 71            MOV M,C
0945 23            INX H
0946 72            MOV M,D
0947 6A            MOV L,D      STORE BIN EXP

                                EXPONENT REDUCTION BY
                                ---------------------
                                MANTISSA SHIFTING
                                -----------------
                                BC - BINARY MANTISSA
                                E - DECIMAL EXPONENT
                                L BINARY EXPONENT
0948 AF      BTD06:XRA A        CLEAR A,H,D
0949 57            MOV D,A      D - MANTISSA SIGN FLAG
094A 67            MOV H,A      H - MANTISSA UNDERFLOW BYTE
094B 78            MOV A,B      LOAD MS MANTISSA BYTE
094C A7            ANA A        TEST
094D F25B09        JP BTD07     SKIP IF MANTISSA POSITIVE
0950 2F            CMA          COMPLEMENT
0951 47            MOV B,A
0952 79            MOV A,C      LOAD LS MANTISSA BYTE
0953 2F            CMA          COMPLEMENT
0954 C601          ADI 01H      INCREMENT
0956 4F            MOV C,A
0957 78            MOV A,B
0958 8C            ADC H        ADD CY, H CLEAR
0959 47            MOV B,A
095A 15            DCR D        SET - SIGN FLAG
095B 7D      BTD07:MOV A,L      LOAD BINARY EXPONENT
095C A7            ANA A        TEST, CLEAR CY
095D CA6D09        JZ BTD09     EXPONENT ZERO, SHIFTING DONE
0960 78      BTD08:MOV A,B      SHIFT MANTISSA RIGHT
0961 1F            RAR          SHIFT MS BYTE
0962 47            MOV B,A
0963 79            MOV A,C
0964 1F            RAR          SHIFT LS BYTE
0965 4F            MOV C,A
0966 7C            MOV A,H
0967 1F            RAR          SHIFT CY INTO UNDERFLOW BYTE
0968 67            MOV H,A
0969 2C            INR L        INCREMENT BINARY EXPONENT,
                                UPDATE ZERO FLAG
096A C26009        JNZ BTD08    REPEAT LOOP IF BINARY
                                EXPONENT STILL NEGATIVE
```

```
                                         SHIFT OUT DECIMAL DIGITS
                                         ------------------------
096D 7A       BTD09:MOV A,D              LOAD MANTISSA SIGN FLAG
096E 54             MOV D,H              STORE MANTISSA UNDERFLOW BYTE
096F 212920         LXI H,DECV1          LOAD DECIMAL DATA POINTER
0972 77             MOV M,A              STORE SIGN BYTE
0973 23             INX H
                                         HL - DATA POINTER
                                         BC - POS BINARY MANTISSA
                                         D - MANTISSA UNDERFLOW BYTE
                                         E - DECIMAL EXPONENT IN
                                             TWOS COMPLEMENT BINARY
0974 D5             PUSH D               STORE DECIMAL EXPONENT
0975 1E05           MVI E,05H            LOAD COUNTER
0977 E5       BTD10:PUSH H               PUSH ADDRESS POINTER
0978 E5             PUSH H               CLEAR TOP OF STACK TO USE HL
                                         AND STACK TOP AS 4 BYTE
                                         WORK SPACE
0979 A7             ANA A                CLEAR CARRY
097A 7A             MOV A,D              LOAD UNDERFLOW BYTE
097B 17             RAL                  SHIFT
097C 6F             MOV L,A
097D 79             MOV A,C              LOAD LS MANTISSA BYTE
097E 17             RAL
097F 67             MOV H,A
0980 E3             XTHL                 SWITCH WORK SPACE BYTES
0981 78             MOV A,B              LOAD MS MANTISSA BYTE
0982 17             RAL
0983 6F             MOV L,A
0984 3E00           MVI A,00H            CLEAR A
0986 17             RAL                  SHIFT CY TO OVERFLOW BYTE
                                         CLEAR CY
0987 67             MOV H,A
0988 CD0F04         CALL FSLFA           SHIFT WS LEFT ONCE MORE
098B E3             XTHL                 GET LS HALF OF WS
098C 7D             MOV A,L              ADD UNSHIFTED BINARY MANTISSA
                                         TO GET X 5 PRODUCT
098D 82             ADD D
098E 6F             MOV L,A
098F 7C             MOV A,H
0990 89             ADC C
0991 67             MOV H,A
0992 E3             XTHL                 GET MS HALF OF WS
0993 7D             MOV A,L
0994 88             ADC B
0995 6F             MOV L,A
0996 7C             MOV A,H
0997 CE00           ACI 00H              ADD CY, CLEAR CY
0999 67             MOV H,A
099A CD0F04         CALL FSLFA           SHIFT WS TO GET X 10 PROD
                                         STORE NEW DATA
099D 45             MOV B,L              STORE MS MANTISSA BYTE
099E 7C             MOV A,H              STORE OVERFLOW BYTE IN A
```

```
099F E3                 XTHL          GET LS HALF OF WS
09A0 4C                 MOV C,H       STORE LS MANTISSA BYTE
09A1 55                 MOV D,L       STORE UNDERFLOW BYTE
09A2 E1                 POP H         RESTORE STACK
09A3 E1                 POP H         GET POINTER ADDRESS

                                      LEADING ZERO SUPPRESSION
                                      ------------------------
09A4 F5                 PUSH PSW      SAVE DEC DIG
09A5 7B                 MOV A,E       LOAD COUNTER
09A6 EE05               XRI 05H       TEST FOR 1ST DIGIT
09A8 C2B609             JNZ BTD11     SKIP IF NOT 1ST DIG
                                      ELSE CONTINUE
09AB F1                 POP PSW       RESTORE DEC DIG, STACK
09AC A7                 ANA A         TEST
09AD C2B709             JNZ BTD12     SKIP IF NON ZERO DIG
                                      ELSE CONT
09B0 E3                 XTHL          LOAD DECIMAL EXPONENT
09B1 2D                 DCR L         DECREMENT DEC EXP
09B2 E3                 XTHL          RESTORE EXP ON STACK
09B3 C37709             JMP BTD10     SHIFT OUT OTHER 1ST DIG
09B6 F1           BTD11:POP PSW       RELOAD DEC DIGIT
                                      CONTINUE
09B7 1D           BTD12:DCR E         DECREMENT COUNTER
09B8 CAC009             JZ BTD13      SKIP STORE IF ZERO

                                      STORE OVERFLOW AS DEC DIGIT
                                      ---------------------------
09BB 77                 MOV M,A       STORE DECIMAL DIGIT
09BC 23                 INX H
09BD C37709             JMP BTD10     REPEAT LOOP FOR NEXT DIGIT

                                      TEST FOR ROUNDUP
                                      ----------------
09C0 D605         BTD13:SUI 05
09C2 FA070A             JM BTD14      SKIP IF LAST DIGIT LT 5
                                      ELSE DO

                                      ROUNDUP
                                      -------
09C5 212D20             LXI H,DECV5   POINT TO LS DIGIT
09C8 7E                 MOV A,M       LOAD
09C9 3C                 INR A         INCREMENT
09CA 27                 DAA
09CB 77                 MOV M,A       STORE
09CC E6F0               ANI 0F0H      MASK FOR CY
09CE CA070A             JZ BTD14      ROUNDUP COMPLETED
                                      ELSE A CY WAS DETECTED
09D1 7E                 MOV A,M       RELOAD LS DIGIT
09D2 E60F               ANI 0FH       MASK OFF CY
09D4 77                 MOV M,A
09D5 2B                 DCX H         SET POINTER TO NEXT DIGIT
09D6 7E                 MOV A,M       LOAD 3RD DIGIT
09D7 3C                 INR A         INCREMENT
```

```
09D8 27              DAA
09D9 77              MOV M,A       STORE
09DA E6F0            ANI 0F0H      MASK FOR CY
09DC CA070A          JZ BTD14      ROUNDUP COMPLETED, ELSE
                                   ANOTHER CY DETECTED

09DF 7E              MOV A,M
09E0 E60F            ANI 0FH       MASK CY OFF 3RD DIGIT
09E2 77              MOV M,A
09E3 2B              DCX H
09E4 7E              MOV A,M       LOAD 2ND DIGIT
09E5 3C              INR A
09E6 27              DAA
09E7 77              MOV M,A
09E8 E6F0            ANI 0F0H
09EA CA070A          JZ BTD14      DONE, ELSE OTHER CY
09ED 7E              MOV A,M
09EE E60F            ANI 0FH       MASK CY OFF 2ND DIGIT
09F0 77              MOV M,A
09F1 2B              DCX H
09F2 7E              MOV A,M
09F3 3C              INR A
09F4 27              DAA
09F5 77              MOV M,A
09F6 E6F0            ANI 0F0H
09F8 CA070A          JZ BTD14      ROUNDUP HOPEFULY COMPLETED
                                   ELSE CY OUT OF MS DIGIT

                                   LOAD 1.000 AND INC EXPONENT
                                   ----------------------------
09FB 3601            MVI M,01H     LOAD 1 TO MS DIGIT
09FD AF              XRA A         CLEAR A
09FE 23              INX H
09FF 77              MOV M,A       CLEAR 2ND DIGIT
0A00 23              INX H
0A01 77              MOV M,A       CLEAR 3RD DIGIT
0A02 23              INX H
0A03 77              MOV M,A       CLEAR LS DIGIT
0A04 D1              POP D         LOAD DECIMAL EXPONENT
0A05 1C              INR E         INCREMENT
0A06 D5              PUSH D        REPLACE ON STACK

                                   CONVERT, UNPACK AND STORE EXP
                                   -----------------------------
0A07 D1         BTD14:POP D        LOAD DECIMAL EXPONENT
0A08 212E20          LXI H,DECV6   POINT AT EXPONENT SIGN
0A0B AF              XRA A         CLEAR A
0A0C 57              MOV D,A       CLEAR EXPONENT SIGN FLAG
0A0D 7B              MOV A,E       LOAD EXPONENT
0A0E 3D              DCR A         DECREMENT EXP TO PLACE
                                   DECIMAL POINT AFTER
                                   LEADING DIGIT
0A0F 5F              MOV E,A       STORE
0A10 A7              ANA A         TEST
0A11 F2180A          JP BTD15      SKIP IF SIGN +, ELSE
```

```
                                  COMPLEMENT EXP, SIGN FLAG
0A14 2F          CMA              COMPLEMENT EXPONENT
0A15 3C          INR A            INCREMENT
0A16 5F          MOV E,A          STORE
0A17 15          DCR D            SET EXP SIGN FLAG TO
                                  FFH FOR - SIGN
0A18 72    BTD15:MOV M,D          STORE EXP SIGN FLAG
0A19 23          INX H
                                  CONVERT EXPONENT
0A1A 0608        MVI B,08H        LOAD COUNTER
0A1C AF          XRA A            CLEAR A
0A1D 4F          MOV C,A          STORE BCD DEC EXP
0A1E 57          MOV D,A          CLEAR D
0A1F 7B    BTD16:MOV A,E          LOAD DEC EXP IN BINARY
0A20 07          RLC              SHIFT IN NEXT BIT
0A21 5F          MOV E,A          STORE BINARY EXP
0A22 79          MOV A,C          LOAD DEC EXP
0A23 8A          ADC D            ADD NEXT BIT
0A24 27          DAA              DECIMAL ADJUST
0A25 DA350A      JC BTD17         OVERFLOW ERROR EXIT
0A28 05          DCR B            DEC COUNTER
0A29 CA3D0A      JZ BTD18         DONE IF ZERO
0A2C 87          ADD A            MULT DEC EXP BY 2
0A2D 27          DAA
0A2E DA350A      JC BTD17         OVERFLOW ERROR EXIT
0A31 4F          MOV C,A          STORE DEC EXP
0A32 C31F0A      JMP BTD16        REPEAT LOOP
0A35 F5    BTD17:PUSH PSW
0A36 E5          PUSH H
0A37 216660      LXI H,6066H      LOAD ERROR CODE 14
0A3A C34000      JMP ERRFA        ERROR EXIT
0A3D 4F    BTD18:MOV C,A          SAVE DEC EXP
0A3E E6F0        ANI 0F0H         MASK FOR MS DIGIT
0A40 0F          RRC              SHIFT RIGHT 4 BITS
0A41 0F          RRC
0A42 0F          RRC
0A43 0F          RRC
0A44 77          MOV M,A          STORE
0A45 23          INX H
0A46 79          MOV A,C
0A47 E60F        ANI 0FH          MASK FOR LS DIGIT
0A49 77          MOV M,A          STORE
0A4A E1    BTD0E:POP H            RESTORE INTERNAL REGISTERS
0A4B D1          POP D
0A4C C1          POP B
0A4D F1          POP PSW
0A4E C9          RET              DONE
```

D. Calculator Function Routines

Forwarding Addresses

```
0C00 C32A0C CENTF:JMP CENTS    LOAD Y REG
0C03 C30F0D CCGSF:JMP CCGSS    CHANGE SIGN
0C06 C3220D CHEXF:JMP CHEXS    DECIMAL TO HEX
0C09 C3500D CSXYF:JMP CSXYS    SWAP X AND Y
0C0C C3820D CMULF:JMP CMULS    MULTIPLY
0C0F C3920D CDIVF:JMP CDIVS    DIVIDE
0C12 C3A20D CADDF:JMP CADDS    ADD
0C15 C3B20D CSUBF:JMP CSUBS    SUBTRACT
0C18 C3F50D CSTRF:JMP CSTRS    STORE
0C1B C32D0E CRCLF:JMP CRCLS    RECALL
0C1E C3490C DENFA:JMP DENST    DECIMAL ENTER
0C21 C3B00C DDSFA:JMP DDSST    DECIMAL DISPLAY
0C24 C3860E FPDSF:JMP FPDSS    F.P. WORKSPACE PUSH
0C27 C3A50E FPDRF:JMP FPDRS    F.P. WORKSPACE POP
```

The Y Register Load Function

```
0C2A CD860E CENTS:CALL FPDSS   PUSH FLOATING POINT WORKSPACE
0C2D AF           XRA A
0C2E 320E20       STA FTNFL    CLEAR FTN FLAG
0C31 CD490C       CALL DENST   MOVE DATA FROM DISP
                               FILE TO DECVN
0C34 CD0008       CALL DTBFA   CONVERT TO BINARY
                               STORE IN FPV2N
0C37 011F20       LXI B,FPV21
0C3A 113420       LXI D,YREG1
0C3D 2603         MVI H,03H
0C3F CD4F00       CALL BKMFA   MOVE ENTERED DATA TO YREG
0C42 CDB00C       CALL DDSST   DISP ENTERED DATA
0C45 CDA50E       CALL FPDRS   POP FLOATING POINT WORKSPACE
0C48 C9           RET
```

The Decimal Data Entry Routine

```
0C49 F5      DENST:PUSH PSW     SAVE INT REGISTERS
0C4A C5            PUSH B
0C4B D5            PUSH D
0C4C E5            PUSH H
0C4D 212920        LXI H,DECV1  POINT TO DECVN
0C50 0608          MVI B,08H
0C52 AF            XRA A        CLEAR A
0C53 77      DEN01:MOV M,A      CLEAR DECVN
0C54 23            INX H
0C55 05            DCR B
0C56 C2530C        JNZ DEN01    REPEAT UNTIL DONE
0C59 211820        LXI H,DDAT7  POINT TO DISPLAY FILE
0C5C 112920        LXI D,DECV1  POINT TO DEC DATA FILE
0C5F 0E04          MVI C,04H    LOAD DIGIT COUNTER
0C61 7E            MOV A,M      LOAD 1ST DIGIT OR SIGN
0C62 E6F0          ANI 0F0H     TEST FOR SIGN
0C64 CA6C0C        JZ DEN02     WAS DIGIT IF ZERO, ELSE
                                STORE - SIGN FLAG
0C67 7E            MOV A,M      RELOAD SIGN FLAG
0C68 12            STAX D       STORE IN DEC DATA FILE
0C69 C3700C        JMP DEN03    GO TO NEXT STEP
0C6C 7E      DEN02:MOV A,M      RELOAD DIGIT
0C6D 13            INX D        POINT TO 1ST DIGIT
0C6E 12            STAX D       STORE DIGIT
0C6F 0D            DCR C        DECREMENT DIGIT COUNTER
0C70 13      DEN03:INX D        POINT TO NEXT DEC DIGIT
0C71 2B            DCX H        POINT TO NEXT INPUT LOCN

                                LOAD DIGITS
                                -----------
0C72 7E      DEN04:MOV A,M      LOAD NEXT CHARACTER
0C73 E6F0          ANI 0F0H     TEST FOR - SIGN
0C75 C2A00C        JNZ DEN06    WAS - SIGN, ELSE
0C78 7E            MOV A,M      RELOAD AND
0C79 EE0E          XRI 0EH      TEST FOR EXP CODE
0C7B CA9A0C        JZ DEN05     WAS EXP CODE
0C7E 7E            MOV A,M      RELOAD AND
0C7F 12            STAX D       STORE NEXT DIGIT
0C80 2B            DCX H        POINT TO NEXT INPUT
0C81 13            INX D        POINT TO NEXT OUTPUT
0C82 0D            DCR C        DECREMENT COUNTER
0C83 C2720C        JNZ DEN04    REPEAT UNTIL 4 DIGITS LOADED

                                LOAD EXP SIGN OR VALUE
                                ----------------------
0C86 7E            MOV A,M      LOAD A SIGN FLAG
```

```
                                OR AN EXPONENT
0C87 E6F0         ANI 0F0H      TEST FOR - SIGN
0C89 C2A00C       JNZ DEN06     WAS - SIGN
0C8C 7E           MOV A,M       RELOAD
0C8D EE0E         XRI 0EH       TEST FOR EXP CODE
0C8F CA9A0C       JZ DEN05      WAS EXP CODE, ELSE
                                IT WAS EXPONENT
0C92 7E           MOV A,M       RELOAD EXPONENT
0C93 113020       LXI D,DECV8   POINT TO LS EXP DIGIT
0C96 12           STAX D        STORE EXPONENT
0C97 C3AB0C       JMP DEN08     GO TO END
                                STORE - EXP FLAG

                                ADDRESSED CHAR IS EXP SIGN
                                --------------------------
0C9A 112E20 DEN05:LXI D,DECV6   POINT TO EXP SIGN
0C9D C3A50C       JMP DEN07
0CA0 112E20 DEN06:LXI D,DECV6   POINT TO EXP SIGN
0CA3 3EFF         MVI A,0FFH    LOAD - SIGN FLAG
0CA5 12     DEN07:STAX D        STORE SIGN FLAG
0CA6 13           INX D         POINT TO LS EXP DIG
0CA7 13           INX D
0CA8 2B           DCX H         POINT TO EXPONENT
0CA9 7E           MOV A,M
0CAA 12           STAX D        STORE IN DECV8
0CAB E1     DEN08:POP H
0CAC D1           POP D
0CAD C1           POP B
0CAE F1           POP PSW
0CAF C9           RET
```

__The__ __Decimal__ __Data__ __Display__ __Routine__

```
0CB0 F5     DDSST:PUSH PSW
0CB1 C5           PUSH B
0CB2 D5           PUSH D
0CB3 E5           PUSH H
0CB4 3EFF         MVI A,0FFH
0CB6 323720       STA CLRFL     SET CLEAR FLAG
0CB9 213020       LXI H,DECV8
0CBC 3E90         MVI A,90H     AUTO INC WRITE DISP RAM
0CBE D305         OUT C8279
0CC0 7E           MOV A,M       LOAD LS EXP DIG
0CC1 2B           DCX H
0CC2 CD4C00       CALL CVSFA    CONVERT
0CC5 D304         OUT D8279     OUTPUT
0CC7 7E           MOV A,M       LOAD MS EXP DIG
0CC8 A7           ANA A         TEST
0CC9 CAD40C       JZ DDS01      SKIP IF ZERO, ELSE
```

```
0CCC F5                 PUSH PSW     DISPLAY ERROR
0CCD E5                 PUSH H
0CCE 21D660             LXI H,60D6H  LOAD ERROR CODE, 15
0CD1 C34000             JMP ERRFA
0CD4 2B           DDS01:DCX H        POINT TO SIGN BYTE
0CD5 7E                 MOV A,M      LOAD AND TEST
0CD6 A7                 ANA A
0CD7 CADC0C             JZ DDS02     SKIP IF ZERO, ELSE
0CDA 3E04               MVI A,04H    LOAD - SIGN CODE
0CDC D304         DDS02:OUT D8279    OUT TO DISP RAM
0CDE 3E97               MVI A,97H    LOAD AND WRITE E
0CE0 D304               OUT D8279
0CE2 2B                 DCX H        POINT TO LS MANT BYTE
0CE3 7E                 MOV A,M      LOAD LS MANT DIG
0CE4 2B                 DCX H
0CE5 CD4C00             CALL CVSFA
0CE8 D304               OUT D8279
0CEA 7E                 MOV A,M      LOAD 3RD MANT DIG
0CEB 2B                 DCX H
0CEC CD4C00             CALL CVSFA
0CEF D304               OUT D8279
0CF1 7E                 MOV A,M      LOAD 2ND MANT DIG
0CF2 2B                 DCX H
0CF3 CD4C00             CALL CVSFA
0CF6 D304               OUT D8279
0CF8 7E                 MOV A,M      LOAD MS MANT DIG
0CF9 CD4C00             CALL CVSFA
0CFC C608               ADI 08H      ADD DECIMAL POINT
0CFE D304               OUT D8279
0D00 2B                 DCX H
0D01 7E                 MOV A,M      LOAD SIGN BYTE
0D02 A7                 ANA A
0D03 CA080D             JZ DDS03     SKIP IF POSITIVE, ELSE
0D06 3E04               MVI A,04H    LOAD AND DISP - SIGN
0D08 D304         DDS03:OUT D8279    DISP SIGN OR BLANK
0D0A E1                 POP H
0D0B D1                 POP D
0D0C C1                 POP B
0D0D F1                 POP PSW
0D0E C9                 RET
```

The Sign Change Function

```
0D0F F5      CCGSS:PUSH PSW      SAVE INTERNAL REGISTERS
0D10 C5            PUSH B
0D11 D5            PUSH D
0D12 E5            PUSH H
0D13 AF            XRA A
0D14 320E20        STA FTNFL     CLEAR FTN FLAG
0D17 0104FF        LXI B,0FF04H  LOAD SIGN FLAG
                                 AND SEVEN SEG CODE
0D1A CD5200        CALL DSPFA
0D1D E1            POP H         RESTORE INTERNAL REGISTERS
0D1E D1            POP D
0D1F C1            POP B
0D20 F1            POP PSW
0D21 C9            RET           DONE
```

The Decimal to Hexadecimal Conversion Function

```
0D22 F5      CHEXS:PUSH PSW      SAVE INT REGISTERS
0D23 C5            PUSH B
0D24 E5            PUSH H
0D25 3EFF          MVI A,0FFH    SET CLEAR FLAG
0D27 323720        STA CLRFL
0D2A CD4900        CALL CLRFA    CLEAR DISP AND RAM
0D2D 0E03          MVI C,03H     LOAD COUNTER
0D2F 213620        LXI H,YREG3   POINT TO EXP OF DATA
0D32 3E90          MVI A,90H     SELECT DISP LOCN 0
0D34 D305          OUT C8279     WITH AUTO INC ON
0D36 46      CFHX1:MOV B,M       LOAD DATA
0D37 78            MOV A,B
0D38 2B            DCX H
0D39 CD4C00        CALL CVSFA    MASK AND CONVERT TO
                                 7 SEG CODE
0D3C D304          OUT D8279     WRITE TO DISP
0D3E 78            MOV A,B
0D3F 0F            RRC           SHIFT RIGHT TO GET MS NIBBLE
0D40 0F            RRC
0D41 0F            RRC
0D42 0F            RRC
0D43 CD4C00        CALL CVSFA    MASK AND CONVERT
0D46 D304          OUT D8279
0D48 0D            DCR C         DECREMENT COUNTER
0D49 C2360D        JNZ CFHX1     REPEAT IF NOT DONE
0D4C E1            POP H
0D4D C1            POP B
0D4E F1            POP PSW
0D4F C9            RET           DONE
```

The X and Y Register Exchange Function

```
0D50 CD860E CSXYS:CALL FPDSS   SAVE F.P. SCRATCHPAD
0D53 AF           XRA A        CLEAR A
0D54 320E20       STA FTNFL    CLEAR FTN FLAG
0D57 013420       LXI B,YREG1  POINT TO Y REG
0D5A 111F20       LXI D,FPV21  POINT TO FPV2N
0D5D 2603         MVI H,03H    LOAD BYTE COUNT
0D5F CD4F00       CALL BKMFA   MOVE YREGN TO FPV2N
0D62 013120       LXI B,XREG1
0D65 113420       LXI D,YREG1
0D68 2603         MVI H,03H
0D6A CD4F00       CALL BKMFA   MOVE XREGN TO YREGN
0D6D 011F20       LXI B,FPV21
0D70 113120       LXI D,XREG1
0D73 2603         MVI H,03H
0D75 CD4F00       CALL BKMFA   MOVE FPV2N TO X REG
0D78 CD0308       CALL BTDFA   CONVERT FPV2N TO DECIMAL
0D7B CDB00C       CALL DDSST   DISPLAY
0D7E CDA50E       CALL FPDRS   RESTORE F.P. SCRATCHPAD
0D81 C9           RET          DONE
```

The Multiplication Function

```
0D82 CD860E CMULS:CALL FPDSS   STORE INT REGS
0D85 CDC20D       CALL CFSTS   ENTER DATA
0D88 CD0004       CALL MULFA   MULTIPLY
0D8B CDE30D       CALL CFENS   STORE AND DISPLAY RESULT
0D8E CDA50E       CALL FPDRS   RESTORE INT REGISTERS
0D91 C9           RET
```

The Division Function

```
0D92 CD860E CDIVS:CALL FPDSS
0D95 CDC20D       CALL CFSTS
0D98 CD0304       CALL DIVFA   DIVIDE
0D9B CDE30D       CALL CFENS
0D9E CDA50E       CALL FPDRS
0DA1 C9           RET
```

The Addition Function

```
0DA2 CD860E CADDS:CALL FPDSS
0DA5 CDC20D       CALL CFSTS
0DA8 CD0604       CALL ADDFA   ADD
0DAB CDE30D       CALL CFENS
0DAE CDA50E       CALL FPDRS
0DB1 C9           RET
```

The Subtraction Function

```
0DB2 CD860E CSUBS:CALL FPDSS
0DB5 CDC20D       CALL CFSTS
0DB8 CD0904       CALL SUBFA   SUBTRACT
0DBB CDE30D       CALL CFENS
0DBE CDA50E       CALL FPDRS
0DC1 C9           RET
```

The Calculator Function Start Routine

```
0DC2 AF        CFSTS:XRA A        CLEAR A
0DC3 320E20          STA FTNFL    CLEAR FTN FLAG
0DC6 CD490C          CALL DENST
0DC9 CD0008          CALL DTBFA
0DCC 011F20          LXI B,FPV21
0DCF 113120          LXI D,XREG1
0DD2 2603            MVI H,03H
0DD4 CD4F00          CALL BKMFA   MOVE NEW DATA TO X REG
0DD7 013420          LXI B,YREG1
0DDA 111C20          LXI D,FPV11
0DDD 2603            MVI H,03H
0DDF CD4F00          CALL BKMFA   MOVE Y REG TO FPV1N
0DE2 C9              RET
```

The Calculator Function End Routine

```
0DE3 011F20    CFENS:LXI B,FPV21  LOAD BLOCKMOVE DATA
0DE6 113420          LXI D,YREG1
0DE9 2603            MVI H,03H
0DEB CD4F00          CALL BKMFA   MOVE RESULT TO YREGN
0DEE CD0308          CALL BTDFA   CONVERT RESULT TO DECIMAL
0DF1 CDB00C          CALL DDSST   DISPLAY RESULT
0DF4 C9              RET
```

The Data Store Function

```
0DF5 F5        CSTRS:PUSH PSW     SAVE INT REGISTERS
0DF6 C5              PUSH B
0DF7 D5              PUSH D
0DF8 E5              PUSH H
0DF9 AF              XRA A        CLEAR A
0DFA 320E20          STA FTNFL    CLEAR FTN FLAG
0DFD 3A1820          LDA DDAT7    LOAD MEMORY ADDRESS
0E00 47              MOV B,A      STORE
0E01 D60A            SUI 0AH      CHECK FOR OUT OF
                                  RANGE ADDRESS
0E03 FA0E0E          JM CSTR1     SKIP IF ADDRESS LT 9,
                                  ELSE ERROR EXIT
0E06 F5              PUSH PSW
0E07 E5              PUSH H
0E08 216060          LXI H,6060H  LOAD ERROR CODE 11
```

```
0E0B C34000        JMP ERRFA
0E0E 78     CSTR1:MOV A,B       LOAD ADDRESS
0E0F 07            RLC          SHIFT TO MULT BY 2
0E10 80            ADD B        ADD TO MULT BY 3
0E11 5F            MOV E,A      STORE OFFSET
0E12 AF            XRA A        CLEAR A
0E13 57            MOV D,A      CLEAR D
0E14 213420        LXI H,YREG1  POINT TO Y REG
0E17 46            MOV B,M      LOAD MS DATA BYTE
0E18 23            INX H
0E19 4E            MOV C,M      LOAD LS BYTE
0E1A 23            INX H
0E1B 7E            MOV A,M      LOAD EXP
0E1C 213820        LXI H,MEM00  LOAD START OF MEMORY FILE
0E1F 19            DAD D        ADD OFFSET
0E20 70            MOV M,B      STORE MS BYTE
0E21 23            INX H
0E22 71            MOV M,C      STORE LS BYTE
0E23 23            INX H
0E24 77            MOV M,A      STORE EXP
0E25 CD4900        CALL CLRFA   CLEAR DISPLAY, DISPLAY FILE
0E28 E1            POP H        RESTORE INT REGISTERS
0E29 D1            POP D
0E2A C1            POP B
0E2B F1            POP PSW
0E2C C9            RET
```

<u>The Data Recall Function</u>

```
0E2D CD860E CRCLS:CALL FPDSS    PUSH F.P. SCRATCHPAD
0E30 AF            XRA A        CLEAR A
0E31 320E20        STA FTNFL    CLEAR FTN FLAG
0E34 3A1820        LDA DDAT7    LOAD MEMORY ADDRESS
0E37 47            MOV B,A      STORE
0E38 D60A          SUI 0AH      CHECK FOR OUT OF
                                RANGE ADDRESS
0E3A FA450E        JM CRCL1     SKIP IF IN RANGE, ELSE ERROR
0E3D F5            PUSH PSW     ERROR EXIT
0E3E E5            PUSH H
0E3F 21B560        LXI H,60B5H  LOAD ERROR CODE 12
0E42 C34000        JMP ERRFA
0E45 78     CRCL1:MOV A,B       LOAD ADDRESS
0E46 07            RLC          SHIFT TO MULT BY 2
0E47 80            ADD B        ADD TO MULT BY 3
0E48 5F            MOV E,A      STORE OFFSET
0E49 AF            XRA A        CLEAR A
0E4A 57            MOV D,A      CLEAR D
0E4B 213820        LXI H,MEM00  LOAD START OF MEM FILE
0E4E 19            DAD D        ADD OFFSET
```

```
0E4F 44              MOV B,H      LOAD BLOCKMOVE DATA
0E50 4D              MOV C,L
0E51 111F20          LXI D,FPV21
0E54 2603            MVI H,03H
0E56 CD4F00          CALL BKMFA   MOVE DATA TO FPV2N
0E59 CD0308          CALL BTDFA   CONVERT
0E5C CDB00C          CALL DDSST   DISPLAY
                                  STORE DATA IN DDATN
0E5F 212920          LXI H,DECV1  SOURCE FILE
0E62 111820          LXI D,DDAT7  DESTINATION FILE
0E65 7E              MOV A,M      LOAD MANT SIGN
0E66 23              INX H        POINT TO MS MANT DIGIT
0E67 A7              ANA A        TEST SIGN
0E68 CA6D0E          JZ CRCL2     SKIP IF ZERO, ELSE
0E6B 12              STAX D       STORE
0E6C 1B              DCX D        NEXT DEST FILE LOCN
0E6D 0604      CRCL2:MVI B,04H    LOAD DIGIT COUNTER, 4D
0E6F 7E        CRCL3:MOV A,M      LOAD DIGIT
0E70 12              STAX D       STORE
0E71 23              INX H        UPDATE POINTERS
0E72 1B              DCX D
0E73 05              DCR B        DECREMENT COUNTER
0E74 C26F0E          JNZ CRCL3    REPEAT
0E77 7E              MOV A,M      LOAD EXP SIGN
0E78 23              INX H        POINT TO LS EXP DIG
0E79 23              INX H
0E7A A7              ANA A        TEST SIGN
0E7B CA800E          JZ CRCL4     SKIP IF POS, ELSE
0E7E 12              STAX D       STORE EXP SIGN
0E7F 1B              DCX D        UPDATE POINTER
0E80 7E        CRCL4:MOV A,M      LOAD EXPONENT DIGIT
0E81 12              STAX D       STORE
0E82 CDA50E          CALL FPDRS   POP F.P. SCRATCHPAD
0E85 C9              RET
```

<u>The Floating Point Workspace Save Routine</u>

```
0E86 F3        FPDSS:DI           DISABLE INTERRRUPTS
0E87 E3              XTHL         LOAD RETURN ADD, PUSH H
0E88 D5              PUSH D
0E89 C5              PUSH B
0E8A F5              PUSH PSW
0E8B 225620          SHLD FPDTS   SAVE RET ADDRESS
0E8E AF              XRA A        CLEAR A,H,L
0E8F 67              MOV H,A
0E90 6F              MOV L,A
0E91 39              DAD SP       LOAD STACK POINTER
0E92 111C20          LXI D,FPV11  DATA START ADDRESS
0E95 060D            MVI B,0DH    NO OF DATA BYTES
```

```
0E97 2B      FPDS1:DCX H         SET SP TO NEXT FREE LOCN
0E98 1A            LDAX D        LOAD DATA
0E99 77            MOV M,A       STORE ON STACK
0E9A 13            INX D         NEXT DATA LOCN
0E9B 05            DCR B         DECREMENT COUNTER
0E9C C2970E        JNZ FPDS1     REPEAT LOOP
0E9F F9            SPHL          LOAD NEW STACK POINTER
0EA0 2A5620        LHLD FPDTS    RELOAD RET ADD
0EA3 FB            EI            REENABLE INTERRUPTS
0EA4 E9            PCHL          RETURN
```

<u>The Floating Point Workspace Restore Routine</u>

```
0EA5 F3      FPDRS:DI            DISABLE INTERUPTS
0EA6 E1            POP H         LOAD RETURN ADDRESS
0EA7 225620        SHLD FPDTS    SAVE RETURN ADD
0EAA AF            XRA A         CLEAR A,H,L
0EAB 67            MOV H,A
0EAC 6F            MOV L,A
0EAD 39            DAD SP        LOAD SP
0EAE 112820        LXI D,FPLCT   LAST DATA ADDRESS
0EB1 060D          MVI B,0DH     NO OF DATA BYTES
0EB3 7E      FPDR1:MOV A,M       LOAD DATA FROM STACK
0EB4 23            INX H         INC SP
0EB5 12            STAX D        STORE DATA
0EB6 1B            DCX D         NEXT DATA LOCATION
0EB7 05            DCR B         DECREMENT COUNTER
0EB8 C2B30E        JNZ FPDR1     REPEAT LOOP
0EBB F9            SPHL          LOAD NEW STACK POINTER
0EBC 2A5620        LHLD FPDTS    RELOAD RET ADD
0EBF F1            POP PSW
0EC0 C1            POP B
0EC1 D1            POP D
0EC2 E3            XTHL          POP HL, STORE RET ADD
0EC3 FB            EI            REENABLE INTERRUPTS
0EC4 C9            RET           RETURN
```

E. Application Module Software

Forwarding Addresses

```
1000 C30C10 MINFA:JMP MINST    MOTOR CONT INIT
1003 C36010 MSTFA:JMP MSTST    MOTOR START
1006 C3CC10 CUPFA:JMP CUPST    MOTOR CONTROL UPDATE
1009 C30A13 MSTPF:JMP MSTPS    STOP MOTOR
```

The Application Module Initialization Routine

```
100C F5      MINST:PUSH PSW    SAVE INT REGISTERS
100D C5            PUSH B
100E D5            PUSH D
100F E5            PUSH H
1010 AF            XRA A       CLEAR A, RUN FLAG
1011 326220        STA RUNFL

                               INIT SCR FIRING TIMER
                               ---------------------
1014 3E32          MVI A,32H   SELECT MODE 1, BINARY
1016 D3FF          OUT TIM1C   WRITE TO TIMER 0
                               NO FIRING PULSES ARE OUTPUT

                               LOAD SET POINTS
                               ---------------
1018 AF            XRA A       CLEAR A
1019 014B10        LXI B,MINDT MOVE DATA FROM
101C 114120        LXI D,MEM03 TABLE TO MEMORY
101F 2615          MVI H,15H   21D BYTES
1021 CD4F00        CALL BKMFA  MOVE

                               DEFAULT UPDATE COUNTER
                               ----------------------
1024 3E0D          MVI A,0DH   LOAD DEFAULT VALUE, 13D
1026 326520        STA ICINV   STORE IN INITIAL VALUE
```

```
                                      SET AUX FTN STEPPING STONES
                                      ---------------------------
1029 210020        LXI H,SSAF0  POINT TO 1ST STEP STONE
102C 23            INX H        POINT TO ADDRESS
102D 010910        LXI B,MSTPF  LOAD MOTOR STOP ADD
1030 71            MOV M,C      STORE ADDRESS
1031 23            INX H
1032 70            MOV M,B
1033 23            INX H
1034 23            INX H
1035 010310        LXI B,MSTFA  POINT TO MOTOR START
1038 71            MOV M,C      STORE ADDRESS
1039 23            INX H
103A 70            MOV M,B
103B 23            INX H
103C 23            INX H
103D 010014        LXI B,LINFF  LINE FREQUENCY MONITOR
1040 71            MOV M,C
1041 23            INX H
1042 70            MOV M,B
1043 CD4900        CALL CLRFA   CLEAR DISPLAY
1046 E1            POP H        RESTORE INT REG
1047 D1            POP D
1048 C1            POP B
1049 F1            POP PSW
104A C9            RET          EXIT

                                DATA TABLE
                                ----------

104B 000000 MINDT:DB 000H,000H,000H CURRENT CONTROL, 0
104E 51EBFB       DB 051H,0EBH,0FBH PROPORTIONAL GAIN, .02
1051 000000       DB 000H,000H,000H INTEGRAL GAIN, 0
1054 7D000A       DB 07DH,000H,00AH RPM SETPOINT, 1000
1057 000000       DB 000H,000H,000H PRESENT RPM, 0
105A 599902       DB 059H,099H,002H CURRENT LIMIT, 2.8
105D 000000       DB 000H,000H,000H PRESENT CURRENT, 0
```

The Motor Start Function

```
1060 F5      MSTST:PUSH PSW     SAVE INT REG
1061 C5            PUSH B
1062 D5            PUSH D
1063 E5            PUSH H
1064 AF            XRA A        CLEAR A,FTN FLAG
1065 320E20        STA FTNFL
                                TEST MAX CURRENT
                                ----------------
1068 3A5020        LDA MEM08    LOAD MS MANT BYTE
106B A7            ANA A        TEST
106C C27610        JNZ MST01    SKIP IF CURRENT ENTERED
                                ELSE ERROR EXIT
106F E5            PUSH H
1070 2160B5        LXI H,0B560H ERROR CODE 21
1073 C34000        JMP ERRFA    ERROR EXIT

                                TEST RPM SETPOINT
                                -----------------
1076 3A4A20 MST01:LDA MEM06
1079 A7            ANA A
107A C28410        JNZ MST02    SKIP IF DATA ENTERED,
                                ELSE ERROR EXIT
107D E5            PUSH H
107E 21B5B5        LXI H,0B5B5H ERROR CODE 22
1081 C34000        JMP ERRFA

                                TEST GAIN COEFICIENTS
                                 --------------------
1084 3A4720 MST02:LDA MEM05     LOAD INTEGRAL GAIN
1087 A7            ANA A
1088 C29910        JNZ MST03    SKIP IF GAIN NON ZERO
                                ELSE TEST PROPORTIONAL GAIN
108B 3A4420        LDA MEM04
108E A7            ANA A
108F C29910        JNZ MST03    SKIP IF NON ZERO, ELSE
                                ERROR EXIT
1092 E5            PUSH H
1093 21F4B5        LXI H,0B5F4H ERROR CODE 23
1096 C34000        JMP ERRFA
                                CLEAR TIME AVE CURRENT
                                 --------------------
1099 AF      MST03:XRA A        CLEAR A, IAVEN
109A 213E20        LXI H,MEM02  POINT TO 1ST BYTE
109D 77            MOV M,A
109E 23            INX H
109F 77            MOV M,A
10A0 23            INX H
10A1 77            MOV M,A
```

```
                                     CLEAR OLD INTEGRAL CONTROL
                                     --------------------------
10A2 215A20          LXI H,CVIC1     POINT TO DESTINATION
10A5 AF              XRA A           CLEAR A,MEMORY
10A6 77              MOV M,A
10A7 23              INX H
10A8 77              MOV M,A
10A9 23              INX H
10AA 77              MOV M,A

                                     TURN ON SCR'S
                                     ------------
10AB 3EFF            MVI A,0FFH      SET MOTOR RUN FLAG
10AD 326220          STA RUNFL
10B0 211F20          LXI H,FPV21     LOAD CONTROL
10B3 3640            MVI M,40H       LOAD STARTING CONTROL
10B5 23              INX H           BINARY 2
10B6 3600            MVI M,00H
10B8 23              INX H
10B9 3602            MVI M,02H
10BB CD6E12          CALL COUTS      OUTPUT CONTROL TO SCR'S
10BE 3EDA            MVI A,0DAH      SET SOD, ENABLE RST 7.5, 5.5
10C0 30              SIM             WRITE MASK
10C1 3A6520          LDA ICINV       LOAD INT COUNTER INIT VALUE
10C4 326420          STA INTCT       INITIALIZE INTERRUPT COUNTER
10C7 E1              POP H
10C8 D1              POP D
10C9 C1              POP B
10CA F1              POP PSW
10CB C9              RET
```

The Motor Control Update Routine

```
10CC CD240C CUPST:CALL FPDSF     SAVE INT REGS,
                                 F.P. WORKSPACE

                                 TEST RPM SETPOINT
                                 -----------------
10CF 3A4A20       LDA MEM06      READ MS MANT BYTE OF
                                 RPM SETPOINT
10D2 A7           ANA A          TEST
10D3 C2DC10       JNZ CUP01      SKIP IF NOT ZERO,ELSE
10D6 CD0A13       CALL MSTPS     STOP MOTOR
10D9 C3E810       JMP CUP0E      GO TO END

                                 READ RPM
                                 --------
10DC CD0614 CUP01:CALL RPMFA     READ RPM, DATA RETURNED
                                 IN FPV2N, MEM07
```

```
10DF CDA611       CALL CCONS  CALCULATE NEW CONTROL VALUE
                              RESULT IN FPV2N

                              OUTPUT CONTROL
                              --------------
10E2 CD6E12       CALL COUTS  OUTPUT CONTROL TO SCR'S

                              TEST CURRENT
                              ------------
10E5 CDF210       CALL TSTIS  TEST FOR OVER CURRENT

10E8 3A6520 CUPOE:LDA ICINV   LOAD INT COUNTER INIT VALUE
10EB 326420       STA INTCT   INITIALIZE INT COUNTER
10EE CD270C       CALL FPDRF  RESTORE INT REG,
                              F.P. SCRATCHPAD
10F1 C9           RET         RETURN FROM INTERRUPT
```

The Current Test Routine

```
10F2 CD0914 TSTIS:CALL RDIFA  READ CURRENT FROM A/D
                              CURRENT IN FPV2N, MEM09

                              TEST INSTANTANEOUS CURRENT
                              --------------------------
10F5 011F20       LXI B,FPV21 MOVE CURRENT
10F8 111C20       LXI D,FPV11 TO DIVIDEND
10FB 2603         MVI H,03H
10FD CD4F00       CALL BKMFA  MOVE
1100 211F20       LXI H,FPV21 POINT TO DEVISOR
1103 3650         MVI M,50H   LOAD BINARY 5
1105 23           INX H
1106 3600         MVI M,00H
1108 23           INX H
1109 3603         MVI M,03H
110B CD0304       CALL DIVFA  DIVIDE INST CURRENT
                              BY SCALING FACTOR
110E 015020       LXI B,MEM08 MOVE MAX CURRENT
1111 111C20       LXI D,FPV11 TO MINUEND
1114 2603         MVI H,03H
1116 CD4F00       CALL BKMFA  MOVE
1119 CD0904       CALL SUBFA  SUBTRACT SCALED CURRENT
                              FROM CURRENT LIMIT
111C 3A1F20       LDA FPV21   LOAD MS MANT BYTE
111F A7           ANA A       TEST
1120 F22D11       JP TSTI1    SKIP IF POSITIVE, ELSE

                              INST OVERCURRENT ERROR
                              ----------------------
1123 CD0A13       CALL MSTPS  STOP MOTOR
```

```
1126 E5            PUSH H
1127 2166B5        LXI H,0B566H ERROR CODE 24
112A C34000        JMP ERRFA

                                TEST AVERAGE CURRENT
                                --------------------
112D 015320 TSTI1:LXI B,MEM09   LOAD INSTANTANEOUS CURRENT
1130 111F20        LXI D,FPV21  TO MULTIPLIER
1133 2603          MVI H,03
1135 CD4F00        CALL BKMFA   MOVE DATA
1138 211C20        LXI H,FPV11  LOAD SCALING FACTOR
113B 3646          MVI M,46H    LOAD BINARY .549 D
113D 23            INX H
113E 3645          MVI M,45H
1140 23            INX H
1141 3600          MVI M,00H
1143 CD0004        CALL MULFA   MULTIPLY, RESULT IN FPV2N
1146 011F20        LXI B,FPV21  MOVE INST CURRENT
1149 111C20        LXI D,FPV11  TO MINUEND
114C 2603          MVI H,03H
114E CD4F00        CALL BKMFA
1151 013E20        LXI B,MEM02  MOVE OLD AVE CURRENT
1154 111F20        LXI D,FPV21  TO SUBTRAHEND
1157 2603          MVI H,03H
1159 CD4F00        CALL BKMFA   MOVE
115C CD0904        CALL SUBFA   SUBTRACT
115F 211C20        LXI H,FPV11  LOAD GAIN CONSTANT
1162 3668          MVI M,68H    LOAD BINARY .0004
1164 23            INX H
1165 36DB          MVI M,0DBH
1167 23            INX H
1168 36F5          MVI M,0F5H
116A CD0004        CALL MULFA   MULTIPLY TO GET OFFSET
116D 013E20        LXI B,MEM02  LOAD OLD AVE CURRENT
1170 111C20        LXI D,FPV11  TO ADDEND
1173 2603          MVI H,03H
1175 CD4F00        CALL BKMFA   MOVE
1178 CD0604        CALL ADDFA   ADD TO GET NEW AVE CURRENT
117B 011F20        LXI B,FPV21  STORE NEW
117E 113E20        LXI D,MEM02  AVERAGE CURRENT
1181 2603          MVI H,03H
1183 CD4F00        CALL BKMFA   MOVE
1186 015020        LXI B,MEM08  LOAD CURRENT LIMIT
1189 111C20        LXI D,FPV11
118C 2603          MVI H,03H
118E CD4F00        CALL BKMFA   MOVE
1191 CD0904        CALL SUBFA   SUBTRACT AVE CURRENT
                                FROM CURRENT LIMIT
1194 3A1F20        LDA FPV21    LOAD MS MANT BYTE OF RESULT
1197 A7            ANA A        TEST
1198 F2A511        JP TSTI2     SKIP IF POS, ELSE
```

```
                              AVE OVERCURRENT EXIT
                              --------------------
119B CD0A13       CALL MSTPS   STOP MOTOR
119E E5           PUSH H
119F 21D6B5       LXI H,0B5D6H LOAD ERROR CODE 25
11A2 C34000       JMP ERRFA
11A5 C9     TSTI2:RET
```

Calculation of Control Output

```
                              PRESENT RPM IN FPV2N, MEM07

                              CALCULATE ERROR
                              ---------------
11A6 014A20 CCONS:LXI B,MEM06  MOVE RPM SETPOINT
11A9 111C20       LXI D,FPV11  TO MINUEND
11AC 2603         MVI H,03H
11AE CD4F00       CALL BKMFA   MOVE
11B1 CD0904       CALL SUBFA   SUBTRACT RPM FROM SETPOINT
11B4 011F20       LXI B,FPV21  SAVE ERROR IN
11B7 115F20       LXI D,CUTR1  TEMPORARY REGISTER
11BA 2603         MVI H,03H
11BC CD4F00       CALL BKMFA
11BF 3A4720       LDA MEM05    LOAD MS BYTE OF INTEGRAL
                               CONTROL GAIN CONSTANT
11C2 A7           ANA A        TEST
11C3 C2D211       JNZ CCON1    SKIP IF NOT ZERO, ELSE

                               ZERO INTEGRAL CONTROL
                               ---------------------
11C6 AF           XRA A        CLEAR A
11C7 215A20       LXI H,CVIC1  POINT TO INTEGRAL CONTROL
11CA 77           MOV M,A
11CB 23           INX H
11CC 77           MOV M,A
11CD 23           INX H
11CE 77           MOV M,A
11CF C31F12       JMP CCON4    CALC PROPORTIONAL CONTROL
11D2 014720 CCON1:LXI B,MEM05  MOVE INTEGRAL GAIN
11D5 111C20       LXI D,FPV11  TO MULTIPLICAND
11D8 2603         MVI H,03H
11DA CD4F00       CALL BKMFA   MOVE
11DD CD0004       CALL MULFA   MULTIPLY
11E0 015A20       LXI B,CVIC1  MOVE CLD CONTROL
11E3 111C20       LXI D,FPV11  TO ADDEND
11E6 2603         MVI H,03H
11E8 CD4F00       CALL BKMFA   MOVE
11EB CD0604       CALL ADDFA   ADD TO GET NEW INT CONT
```

```
                               CLIP INTEGRAL CONTROL VALUE
                               ---------------------------
11EE 3A2120       LDA FPV23    LOAD EXPONENT
11F1 D602         SUI 02H      SUBTRACT 2
11F3 FA0412       JM CCON2     SKIP IF CONTROL L.T. 2.0,
11F6 211F20       LXI H,FPV21  ELSE SET CONTROL TO 2.0
11F9 3640         MVI M,40H
11FB 23           INX H
11FC 3600         MVI M,00H
11FE 23           INX H
11FF 3602         MVI M,02H
1201 C31412       JMP CCON3    CONTINUE
1204 3A1F20 CCON2:LDA FPV21    LOAD MANTISSA
1207 A7           ANA A        TEST
1208 F21412       JP CCON3     SKIP IF MANT POS, ELSE
120B 211F20       LXI H,FPV21  ZERO CONTROL OUTPUT
120E AF           XRA A        CLEAR A
120F 77           MOV M,A
1210 23           INX H
1211 77           MOV M,A
1212 23           INX H
1213 77           MOV M,A
1214 011F20 CCON3:LXI B,FPV21  STORE NEW CONTROL
1217 115A20       LXI D,CVIC1  IN OLD CONTROL
121A 2603         MVI H,03H
121C CD4F00       CALL BKMFA   MOVE

                               PROPORTIONAL CONTROL
                               --------------------
121F 3A4420 CCON4:LDA MEM04    LOAD MS MANT BYTE OF
                               PROP GAIN CONSTANT
1222 A7           ANA A        TEST
1223 CA5712       JZ CCON5     IF ZERO, OUTPUT INT CONT ONLY
                               ELSE CALC PROP CONTROL
1226 015F20       LXI B,CUTR1  MOVE ERROR
1229 111C20       LXI D,FPV11  TO MULTIPLICAND
122C 2603         MVI H,03H
122E CD4F00       CALL BKMFA   MOVE
1231 014420       LXI B,MEM04  MOVE PROP GAIN
1234 111F20       LXI D,FPV21  TO MULTIPLIER
1237 2603         MVI H,03H
1239 CD4F00       CALL BKMFA   MOVE
123C CD0004       CALL MULFA   CALC PROP CONTROL, IN FPV2N

                               ADD PROP PLUS INT CONT
                               ----------------------
123F 3A5A20       LDA CVIC1    LOAD MS MANT BYTE OF
                               INTEGRAL CONTROL
1242 A7           ANA A        TEST
1243 CA6212       JZ CCONE     NO NEED TO ADD SINCE ZERO
1246 015A20       LXI B,CVIC1  MOVE INTEGRAL CONTROL
1249 111C20       LXI D,FPV11  TO ADDEND
124C 2603         MVI H,03H
```

```
124E CD4F00        CALL BKMFA   MOVE
1251 CD0604        CALL ADDFA   ADD CCNTROL OUTPUTS
1254 C36212        JMP CCONE    RETURN TO CONTROL OUTPUT
1257 015A20 CCON5:LXI B,CVIC1   MOVE INT CONTROL
125A 111F20        LXI D,FPV21  TO OUTPUT LOCN
125D 2603          MVI H,03H
125F CD4F00        CALL BKMFA   MOVE
                                STORE CONTROL IN MEMORY REG
1262 011F20 CCONE:LXI B,FPV21   MOVE CONTROL FROM FPV2N
1265 114120        LXI D,MEM03  TO MEMORY
1268 2603          MVI H,03H
126A CD4F00        CALL BKMFA   MOVE DATA
126D C9            RET
```

Output of Control Voltage

```
126E 211F20 COUTS:LXI H,FPV21   LOAD CONTROL VALUE
1271 46            MOV B,M      LOAD MS MANT BYTE
1272 23            INX H
1273 4E            MOV C,M
1274 23            INX H
1275 56            MOV D,M      LOAD EXPONENT

                                TEST FOR NEGATIVE CONTROL
                                -------------------------
1276 78            MOV A,B      LOAD MS MANT BYTE
1277 A7            ANA A        TEST
1278 FA7E12        JM COUT1     SKIP TO NEG OUTPUT IF MINUS
127B C28412        JNZ COUT2    CONT IF POS, ELSE

                                NEGATIVE CONTROL OUTPUT
                                -----------------------
127E 21FFFF COUT1:LXI H,0FFFFH  LOAD MAX COUNT
                                FOR ZERO OUTPUT
1281 C3DF12        JMP COUTE    GO TO END

                                TEST FOR CONTROL G.T.E. 2.0
                                ---------------------------
1284 7A     COUT2:MOV A,D       LOAD EXPONENT
1285 3D            DCR A        DECREMENT
1286 CA9212        JZ COUT3     SKIP IF EXP WAS 1
1289 FA9C12        JM COUT4     SKIP IF EXP L.T.E. 0.0, ELSE

                                CONTROL G.T.E. 2.0
                                ------------------
128C 210100        LXI H,0001H  LOAD MIN COUNT TO
                                GIVE MAX OUTPUT
128F C3DF12        JMP COUTE    GO TO END
```

```
                                EXP WAS 1, SHIFT LEFT
                                ---------------------
1292 A7     COUT3:ANA A         CLEAR CARRY
1293 79           MOV A,C
1294 17           RAL           SHIFT LS BYTE
1295 4F           MOV C,A
1296 78           MOV A,B
1297 17           RAL           SHIFT MS BYTE
1298 47           MOV B,A
1299 C3AB12       JMP COUT5     CONVERT TO COUNT

                                EXP L.T.E. 0
                                ------------
129C 7A     COUT4:MOV A,D       LOAD EXPONENT
129D A7           ANA A         TEST, CLEAR CY
129E CAAB12       JZ COUT5      CONVERT TO COUNT, ELSE
                                SHIFT RIGHT
12A1 78           MOV A,B
12A2 1F           RAR           SHIFT MS BYTE
12A3 47           MOV B,A
12A4 79           MOV A,C
12A5 1F           RAR           SHIFT LS BYTE
12A6 4F           MOV C,A
12A7 14           INR D         INCREMENT EXPONENT
12A8 C39C12       JMP COUT4     RETEST EXPONENT

                                CONVERT CONTROL TO COUNT
                                ------------------------
                                BC - LIMITED INTEGER COUNT
12AB 21EA12 COUT5:LXI H,EXTBL   POINT TO 1ST TABLE ENTRY
12AE AF           XRA A         CLEAR A,D
12AF 57           MOV D,A
12B0 78           MOV A,B       LOAD MS CONTROL BYTE
12B1 E6E0         ANI 0E0H      MASK FOR MS 3 BITS
12B3 0F           RRC           MULT BY 4 TO GET
12B4 0F           RRC           ADDRESS OFFSET
12B5 0F           RRC
12B6 5F           MOV E,A       LOAD TO OFFSET
12B7 19           DAD D         SELECT DATA
12B8 56           MOV D,M       LOAD BASE COUNT
12B9 23           INX H
12BA 5E           MOV E,M
12BB 23           INX H
12BC D5           PUSH D        STORE ON STACK
12BD 56           MOV D,M       LOAD CONSTANT
12BE 23           INX H
12BF 5E           MOV E,M
```

```
                                      FORM SCALING FRACTION
                                      ---------------------
12C0 2603          MVI H,03H          LOAD LOOP COUNTER
12C2 A7      COUT6:ANA A              CLEAR CY
12C3 79            MOV A,C
12C4 17            RAL                SHIFT LS BYTE
12C5 4F            MOV C,A
12C6 78            MOV A,B
12C7 17            RAL                SHIFT MS BYTE
12C8 47            MOV B,A
12C9 25            DCR H              DECREMENT COUNTER
12CA C2C212        JNZ COUT6          REPEAT UNTIL DONE
12CD CD1504        CALL M16BF         MULTIPLY, RESULT IN HL, DE
                                      SCALING FACTOR IN HL
12D0 AF            XRA A              CLEAR A,B
12D1 47            MOV B,A
12D2 7C            MOV A,H
12D3 2F            CMA                COMPLEMENT MS BYTE
12D4 67            MOV H,A
12D5 7D            MOV A,L
12D6 2F            CMA                COMPLEMENT LS BYTE
12D7 C601          ADI 01H            INCREMENT
12D9 6F            MOV L,A
12DA 7C            MOV A,H
12DB 88            ADC B              ADD CY TO MS BYTE
12DC 67            MOV H,A
12DD D1            POP D              LOAD BASE COUNT
12DE 19            DAD D              SUB OFFSET FROM BASE COUNT
                                      TIMER COUNT NOW IN HL
12DF 3E32    COUTE:MVI A,32H          SELECT MODE 1, BINARY
12E1 D3FF          OUT TIM1C          WRITE TO TIMER 0
12E3 7D            MOV A,L            LOAD LS TIMER BYTE
12E4 D3FC          OUT TIM10          OUTPUT
12E6 7C            MOV A,H            LOAD MS TIMER BYTE
12E7 D3FC          OUT TIM10
12E9 C9      COUTX:RET

                                      EXTRAPOLATION TABLE
                                      -------------------


12EA 208D077D EXTBL:DB 020H,08DH,007H,07DH
12EE 1911035D       DB 019H,011H,003H,05DH
12F2 15B402CE       DB 015H,0B4H,002H,0CEH
12F6 12E6029F       DB 012H,0E6H,002H,09FH
12FA 104702A0       DB 010H,047H,002H,0A0H
12FE 0DA702CD       DB 00DH,0A7H,002H,0CDH
1302 0ADA035D       DB 00AH,0DAH,003H,05DH
1306 077D077D       DB 007H,07DH,007H,07DH
```

The Motor Stop Routine

```
130A F5      MSTPS:PUSH PSW      SAVE INT REGS
130B C5            PUSH B
130C D5            PUSH D
130D E5            PUSH H
130E AF            XRA A         CLEAR A, FTN FLAG
130F 320E20        STA FTNFL
1312 3E32          MVI A,32H     SELECT MODE 1, BINARY
1314 D3FF          OUT TIM1C     INITIALIZE TIMER 0
1316 3E0E          MVI A,0EH     MASK RST 7.5, 6.5
1318 30            SIM           WRITE INTERRUPT MASK
1319 AF            XRA A         CLEAR A, MOTOR RUN FLAG
131A 326220        STA RUNFL
131D E1            POP H
131E D1            POP D
131F C1            POP B
1320 F1            POP PSW
1321 C9            RET
```

Forwarding Addresses

```
1400 C30C14 LINFF:JMP LINFS     LINE FREQUENCY
1403 C37E14 RPMIF:JMP RPMIS     RPM PULSE INT
1406 C3D314 RPMFA:JMP RPMST     CALC RPM
1409 C32E15 RDIFA:JMP RDIST     READ CURRENT
```

Line Frequency Monitoring Routine

```
140C F5      LINFS:PUSH PSW
140D C5            PUSH B
140E D5            PUSH D
140F E5            PUSH H
1410 CD4900        CALL CLRFA    CLEAR DISP, FTN FLAG
1413 F3            DI            DISABLE INTERRUPTS
1414 CD6314        CALL LINFW    WAIT FOR RISING EDGE
                                 OF ZCD PULSE
1417 3E70          MVI A,70H     SELECT COUNTER 1, MODE 0
1419 D3FF          OUT TIM1C
141B 3EFF          MVI A,0FFH
141D D3FD          OUT TIM11
141F D3FD          OUT TIM11     START TIMER
1421 0614          MVI B,14H     LOAD 20D TO PULSE COUNTER
```

```
1423 CD6314 LINF1:CALL LINFW    WAIT FOR RISING EDGE
                                OF ZCD PULSE
1426 05             DCR B       DECREMENT PULSE COUNTER
1427 C22314         JNZ LINF1   REPEAT 20 TIMES

                                READ TIMER
                                ----------
142A 3E40           MVI A,40H   LATCH TIMER 1
142C D3FF           OUT TIM1C
142E 212720         LXI H,SHFT5 POINT TO NORM ROUTINE DATA
1431 360F           MVI M,0FH   LOAD EXPONENT, 15D
1433 2B             DCX H
1434 AF             XRA A       CLEAR A
1435 77             MOV M,A     CLEAR 2 LS BYTES
1436 2B             DCX H
1437 77             MOV M,A
1438 2B             DCX H
1439 DBFD           IN TIM11    READ LS COUNT BYTE
143B 2F             CMA         COMPLEMENT
143C 77             MOV M,A     STORE
143D 2B             DCX H
143E DBFD           IN TIM11    READ MS COUNT BYTE
1440 2F             CMA
1441 77             MOV M,A
1442 AF             XRA A       CLEAR A, SIGN FLAG
1443 322220         STA SIGNF
1446 CD0C04         CALL NRMFA  NORMALIZE, DATA NOW IN FPV2N
1449 211C20         LXI H,FPV11 POINT TO DIVIDEND
144C 367A           MVI M,7AH   LOAD BINARY 1.0 E 6
144E 23             INX H
144F 3612           MVI M,12H
1451 23             INX H
1452 3614           MVI M,14H
1454 CD0304         CALL DIVFA  DIVIDE 1.0 E 6 BY PERIOD
1457 CD0308         CALL BTDFA  CONVERT TO DECIMAL
145A CD210C         CALL DDSFA  DISPLAY RESULT
145D FB             EI          REENABLE INTERRUPTS
145E E1             POP H       RESTORE INT REGS
145F D1             POP D
1460 C1             POP B
1461 F1             POP PSW
1462 C9             RET         RETURN

                                WAIT FOR RISING ZCD EDGE
                                ------------------------
1463 3E05   LINFW:MVI A,05H     DELAY LOOP
1465 3D     LFW01:DCR A
1466 C26514       JNZ LFW01
1469 DBF1   LFW02:IN FFDAT       READ FLOP FLOPS
146B E608         ANI 08H       MASK FOR ZCD BIT
146D CA6914       JZ LFW02      WAIT FOR HIGH REGION
1470 3E05         MVI A,05H     DELAY LOOP
1472 3D     LFW03:DCR A
1473 C27214       JNZ LFW03
```

```
1476 DBF1    LFW04:IN FFDAT      READ FLOP FLOPS
1478 E608          ANI 08H       MASK FOR ZCD BIT
147A C27614        JNZ LFW04     WAIT FOR INVERTED
                                 RISING EDGE
147D C9            RET           EDGE DETECTED, RETURN
```

Optical Tachometer Interrupt Service Routine

```
147E F5      RPMIS:PUSH PSW      SAVE INT REGS
147F C5            PUSH B
1480 D5            PUSH D
1481 E5            PUSH H
1482 3ECE          MVI A,0CEH    SET SOD, MASK RST 6.5, 7.5
1484 30            SIM           WRITE MASK
1485 FB            EI            ENABLE INTERRUPTS
1486 3A6220        LDA RUNFL     LOAD RUN FLAG
1489 A7            ANA A         TEST
148A CACE14        JZ RPMIX      EXIT IF NOT SET
148D 3A6420        LDA INTCT     LOAD INTERRUPT COUNTER
1490 3D            DCR A         DECREMENT
1491 326420        STA INTCT     STORE
1494 C2BA14        JNZ RPMI3     SKIP IF NOT ZERO, ELSE

                                 UPDATE CONTROL
                                 --------------
1497 DBF1          IN FFDAT      READ FLIP FLOP DATA
1499 E604          ANI 04H       MASK FOR COUNT OVERRUN BIT
149B C2A714        JNZ RPMI1     SKIP IF VALID, ELSE

                                 RPM BELOW LOWER LIMIT
                                 --------------------
149E 21FFFF        LXI H,0FFFFH  LOAD 0 RPM COUNT
14A1 225820        SHLD RPMCT    STORE
14A4 C3B414        JMP RPMI2     CONTINUE
14A7 3E80    RPMI1:MVI A,80H     LATCH COUNTER
14A9 D3FF          OUT TIM1C     WRITE TO COUNTER 2
14AB DBFE          IN TIM12      LOAD LS COUNT BYTE
14AD 6F            MOV L,A
14AE DBFE          IN TIM12      IN MS BYTE
14B0 67            MOV H,A
14B1 225820        SHLD RPMCT    STORE COUNT
14B4 CD0610  RPMI2:CALL CUPFA    UPDATE CONTROL OUTPUT
14B7 C3CA14        JMP RPMIE     GO TO END
14BA 3D      RPMI3:DCR A         DECREMENT TO TEST FOR 1
14BB C2CA14        JNZ RPMIE     GO TO END IF NOT 1, ELSE
14BE 3EB0          MVI A,0B0H    RESET RPM COUNTER
14C0 D3FF          OUT TIM1C     OUTPUT TO TIMER 2
14C2 3EFF          MVI A,0FFH    LOAD COUNT START
14C4 D3FE          OUT TIM12
```

```
14C6 D3FE          OUT TIM12     START TIMER
14C8 D3F3          OUT TRS12     RESET OVERRUN FLAG
14CA 3E5A   RPMIE:MVI A,5AH      CLEAR SOD, RST 7.5
14CC 30            SIM           SET MASK
14CD FB            EI            ENABLE INTERRUPTS
14CE E1     RPMIX:POP H          RESTORE INT REG
14CF D1            POP D
14D0 C1            POP B
14D1 F1            POP PSW
14D2 C9            RET
```

RPM Reading and Conversion

```
14D3 2A5820 RPMST:LHLD RPMCT   LOAD RPM COUNT
14D6 3EFF          MVI A,0FFH  LOAD TEST BYTE
14D8 BC            CMP H       TEST MS BYTE
14D9 C2F414        JNZ RPM01   SKIP IF COUNT NON ZERO,
                               ELSE TEST LS BYTE
14DC BD            CMP L       ELSE TEST LS BYTE
14DD C2F414        JNZ RPM01   SKIP TO VALID COUNT ROUTINE,
14E0 AF            XRA A       ELSE LOAD ZERO RPM
14E1 211F20        LXI H,FPV21 CLEAR FPV2N
14E4 77            MOV M,A
14E5 23            INX H
14E6 77            MOV M,A
14E7 23            INX H
14E8 77            MOV M,A
14E9 214D20        LXI H,MEM07 ZERO RPM IN MEMORY
14EC 77            MOV M,A
14ED 23            INX H
14EE 77            MOV M,A
14EF 23            INX H
14F0 77            MOV M,A
14F1 C32D15        JMP RPM03   EXIT
14F4 7C     RPM01:MOV A,H      LOAD MS BYTE
14F5 2F            CMA         ONES COMPLEMENT
14F6 57            MOV D,A     STORE IN DE
14F7 7D            MOV A,L
14F8 2F            CMA
14F9 C608          ADI 08H     ADD 08 TO COMPENSATE FOR
                               COUNTER LATCHING OFFSET
14FB 5F            MOV E,A
14FC D20015        JNC RPM02   SKIP IF NO CARRY, ELSE
14FF 14            INR D       INCREMENT MS BYTE
1500 212720 RPM02:LXI H,SHFT5  POINT TO EXP
1503 3617          MVI M,17H   LOAD EXPONENT, 23D
1505 2B            DCX H
1506 AF            XRA A       CLEAR A
1507 77            MOV M,A
```

```
1508 2B             DCX H
1509 73             MOV M,E
150A 2B             DCX H
150B 72             MOV M,D
150C 2B             DCX H
150D 77             MOV M,A      LOAD MS BYTE
150E 322220         STA SIGNF    CLEAR SIGN FLAG
1511 CD0C04         CALL NRMFA   NORMALIZE TIME COUNT
1514 211C20         LXI H,FPV11  LOAD CONSTANT
1517 3672           MVI M,72H    LOAD BINARY 7.5 E 6
                                 TO DIVIDEND
                                 FOR 8 PULSES PER REV
1519 23             INX H
151A 3672           MVI M,72H
151C 23             INX H
151D 3617           MVI M,17H
151F CD0304         CALL DIVFA   DIVIDE TO GET RPM
1522 011F20         LXI B,FPV21  MOVE RPM TO MEMORY
1525 114D20         LXI D,MEM07
1528 2603           MVI H,03H
152A CD4F00         CALL BKMFA   MOVE DATA
152D C9      RPM03:RET           EXIT
```

Reading and Conversion of SCR Current

```
152E D3F4    RDIST:OUT LATSH     LATCH SAMPLE AND HOLD
1530 3E04          MVI A,04H     DELAY
1532 3D      RDI01:DCR A
1533 C23215        JNZ RDI01     LOOP
1536 D3F0          OUT ADC01     START CONVERSION
1538 3E02          MVI A,02H     DELAY FOR EOC PULSE TO FALL
153A 3D      RDI02:DCR A
153B C23A15        JNZ RDI02     LOOP

                                 WAIT FOR END OF CONV
                                 --------------------
153E DBF1    RDI03:IN FFDAT      READ FFDATA
1540 E601          ANI 01H       MASK FOR EOC BIT
1542 CA3E15        JZ RDI03      WAIT
1545 3E0F          MVI A,0FH     WAIT FOR OPTO ISOLATORS
                                 TO SETTLE
1547 3D      RDI04:DCR A
1548 C24715        JNZ RDI04     LOOP
154B DBF0          IN ADC01      READ CONVERTER DATA
154D 212320        LXI H,SHFT1   POINT TO NORM DATA
1550 0600          MVI B,00H     CLEAR B
1552 70            MOV M,B       CLEAR MS BYTE
1553 23            INX H
1554 77            MOV M,A       LOAD DATA
```

```
1555 23           INX H
1556 70           MOV M,B
1557 23           INX H
1558 70           MOV M,B      LOAD LS BYTE
1559 23           INX H
155A 3607         MVI M,07H    LOAD EXPONENT, 7D
155C AF           XRA A        CLEAR A, SIGN FLAG
155D 322220       STA SIGNF
1560 CD0C04       CALL NRMFA   NORMALIZE
1563 211C20       LXI H,FPV11  LOAD MULTIPLICAND
1566 3652         MVI M,52H    LOAD BINARY 10.3 D
1568 23           INX H
1569 3666         MVI M,66H
156B 23           INX H
156C 3604         MVI M,04H
156E CD0004       CALL MULFA   MULTIPLY
1571 011F20       LXI B,FPV21  LOAD RESULT TO MEM REG
1574 115320       LXI D,MEM09
1577 2603         MVI H,03H
1579 CD4F00       CALL BKMFA   MOVE
157C C9           RET          EXIT
            END
```